CLASSIFICATION AND SIGNAL PROCESSING OF RADIO BACKSCATTER FROM METEORS

By

Jared Klemm, B.S.

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electrical Engineering

University of Alaska Fairbanks

December 2019

APPROVED:

Dr. Denise Thorsen, Committee Chair
Dr. Katrina Bossert, Committee Member
Dr. Richard Collins, Committee Member
Dr. Charlie Mayer, Committee Member
Dr. Richard Wies, Chair
Department of Electrical and Computer Engineering
Dr. Bill Schnabel, Dean
College of Engineering and Mines
Dr. Michael Castellini, *Dean of the Graduate School*

Abstract

Ground-based radar systems are routinely used to detect the trails of ionized particles that are formed by meteoroids falling through Earth's atmosphere. The most common use for these meteor radar systems is for atmospheric wind studies of the mesosphere and lower thermosphere (80-100 km altitude). Because these meteor trails are embedded in the background winds of the middle atmosphere, atmospheric winds in that region can be measured by observing the radial velocities of the trails. There has also been a considerable amount of research over the last few decades into estimation of neutral atmospheric temperatures using the measured decay time of meteor trails. Several methods exist for estimating atmospheric temperature using meteor radar observations, but there are limitations to these approaches. This thesis focuses on examining aspects of meteor radar signal and data processing, specifically interferometry and echo classification. Interferometry using the measured signal phase differences between antennas allows for the location of meteor trails to be unambiguously determined. Classification schemes are used to identify which echoes can be modeled as underdense meteors, overdense meteors, or other potentially non-meteor echoes. Finally, based on the proposed classification scheme, this thesis examines several temperature estimation methods for both underdense and overdense echoes and discusses the current issues in this area. Preliminary results from a newly installed meteor radar at Poker Flat Research Range are also presented.

Table of Contents

List of Figures

List of Tables

List of Appendices

Page

Acknowledgments

I am grateful to my advisor, Dr. Denise Thorsen, for her support, guidance, and patience. She has been incredibly supportive of my research endeavors. I am also thankful for support and advice from many faculty: Dr. Richard Collins, Dr. Katrina Bossert, Dr. Charlie Mayer, and fellow graduate students Jennifer Alspach and Jintai Li at University of Alaska Fairbanks. I'd also like to thank Angie Wohlford for going above and beyond to assist and support graduate students in the ECE department.

I am very appreciative of everyone that provided data that allowed me to conduct the research presented in this thesis. Dr. Alexander Kozlovsky provided meteor radar data from Sodankylä Geophysical Observatory, and brought attention to current issues in meteor echo classification. Dr. Richard Collins, Dr. Bifford Williams, Jintai Li, and Jennifer Alspach provided lidar data from Poker Flat Research Range, giving context to my work with meteor radar temperature estimates. This work was supported in part by an NSF grant AGS-1651464.

Lastly, thanks to my wife Tracey, parents Jeff and Cathy, and sister Jessica for their love and support.

List of Abbreviations

AOA angle of arrival

CCD complementary cumulative distribution

CEV confirmed event

MLT mesosphere-lower thermosphere

MPD meteor position data

PFMR Poker Flat Meteor Radar

PFRR Poker Flat Research Range

PMSE polar mesospheric summer echo

SGO Sodankylä Geophysical Observatory

SKiYMET all-sky interferometric meteor radar

SNR signal to noise ratio

SRWTL sodium resonance wind temperature lidar

SVM support vector machine

Chapter 1: Introduction

1.1 Thesis Objectives

This thesis has two primary objectives: to evaluate and compare signal processing techniques for modern meteor radar systems, and to investigate atmospheric temperature estimation techniques that utilize meteor radar data. Most modern meteor radar systems have antenna arrays with an all-sky antenna pattern, so interferometry must be used to determine the location of radar targets. Some interferometry methods for meteor radar systems are evaluated and compared in this thesis. A second challenge in meteor radar signal processing is correctly identifying radar backscatter signals from meteors. There are three classification methods that are compared in this thesis. In addition to these basic signal processing topics, there are several techniques that can be used to estimate neutral atmospheric temperature using meteor radar data. Several of these temperature estimation methods are compared in this thesis using data from other instruments for comparison.

1.2 History of meteor observations

It is estimated that over the course of a day somewhere between one and two hundred million meteoroids create luminescent trails, or meteors, in the Earth's atmosphere. While humans have been observing meteors since at least 1809 B.C. [*McKinley*, 1961], it is only since the 18th century that meteors garnered scientific interest. Several nineteenth-century astronomers took an interest in where meteors come from and what they might be made of. Simultaneous observations separated by several kilometers by Brandes and Benzenberg in 1798 demonstrated that visible meteors were approximately 95 km above the Earth's surface. Public interest in meteors began to grow following the Leonid shower of 1833, when many bright meteor trails were visible over most of North America. Following this shower, Denison Olmstead and others began more structured scientific investigation into meteors [*McKinley*, 1961]. Early studies were mostly focused on meteor counts and shower radiants. Visual

methods for meteor observations remained popular until the early twentieth century, when photographic and radio methods started to be developed [*McKinley*, 1961].

Many early radio observations of meteors were carried out from the reference point of a radio scientist, in the interest of studying radio wave propagation through the atmosphere, while later radio studies of meteors would focus on meteor astronomy or atmospheric dynamics. Radio observations of meteors would not be used widely for meteor astronomy until the mid-twentieth century, following the second World War. In a manner similar to the 1833 Leonids, the 1946 Giacobinids helped to ignite renewed scientific interest in meteor observations. This shower was observed by several teams across North America and Europe, mostly using wartime radars that had been adapted for this purpose. This shower was also the first time that meteor velocity measurements were made using radio observations of meteors [*McKinley*, 1961]. In the mid-twentieth century, radio observations of meteors would be increasingly studied not only from a meteor astronomy viewpoint, but also with the intent of studying the upper atmosphere (the region now commonly referred to as the Mesosphere-Lower Thermosphere, or MLT). Modern radar systems are improving detection of meteors and making more accurate observations. This data is being combined with data from other sources such as rockets, satellites, and lidar facilities to create more holistic observations of the middle and upper atmosphere.

Modern "meteor radar" systems are typically used to observe the velocity of meteors, and use that information to estimate neutral winds in the Mesosphere-Lower Thermosphere region. This work has become so routine that almost all new meteor radar installations are designed by one of two radar companies. These standard radar systems are designed to be unmanned and run as autonomously as possible. These systems come with the advantage of pre-built software packages that take care of meteor detection, and signal processing to estimate neutral winds. Although many of the meteor radars in current operation are of this type, there is still ongoing research into other possible uses besides neutral wind estimation. It is possible to estimate the ambipolar diffusion (diffusion of positive and negative ions in

an electric field) constant from the decay time of meteor backscatter, which could allow for other parameters such as temperature or density to be estimated. For example, there are model relationships between ambipolar diffusion coefficient and neutral temperature, density, and pressure that may allow for these to be estimated as well.

As a meteoroid enters the Earth's atmosphere, friction between the meteoroid's surface and atmospheric particles results in heating of the meteoroid and vaporizing of atoms from its surface. The energy of the collisions between the meteoroid and the atmosphere is large enough to separate electrons from atoms, resulting in a large number of free electrons in the meteor trail. The density of these free electrons will determine how the meteor trail is seen by radio methods. If the electron density is low enough that each electron acts as an individual scatterer, the meteor trail is "underdense." However, if the electron density is high enough that the radar beam cannot penetrate the meteor trail and scatters off of the edge, the meteor trail is "overdense." This classification is a function of both the electron density of the plasma in the meteor trail and the frequency of the observing radar. As such, meteor trails are not inherently underdense or overdense, as a meteor trail observed by one radar may be underdense while the same trail observed by another radar operating at another frequency may be seen as overdense. Figures 1.1 and 1.2 show typical received power waveforms for underdense and overdense meteor trails. The underdense trail will exhibit a rapid increase in received power, followed by an immediate exponential decay in power. The overdense trail will exhibit a rapid increase in received power, followed by a short period of constant power before a rapid loss in received power [*McKinley*, 1961]. Oscillations in received power are also common in overdense meteor echoes, which may be caused by alternating constructive and destructive interference as the trail expands and reflections from different parts of the trail interact.

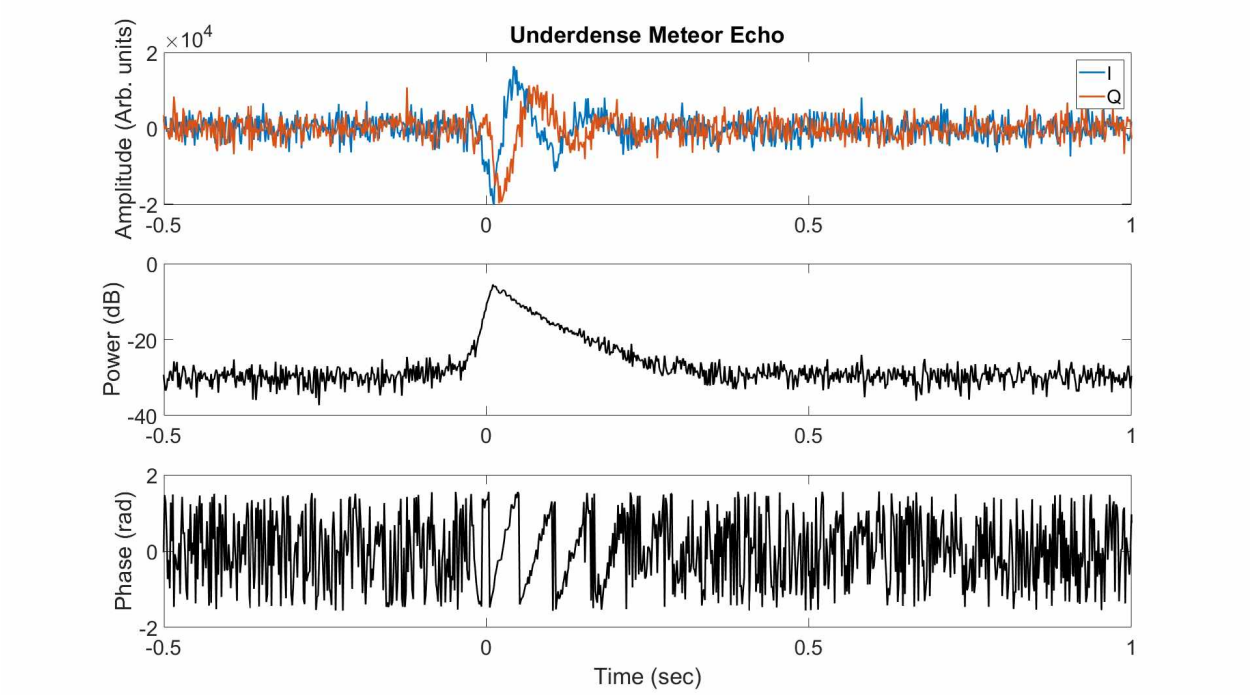

Figure 1.1: Example of an underdense meteor trail echo, exhibiting characteristic behavior in received power.

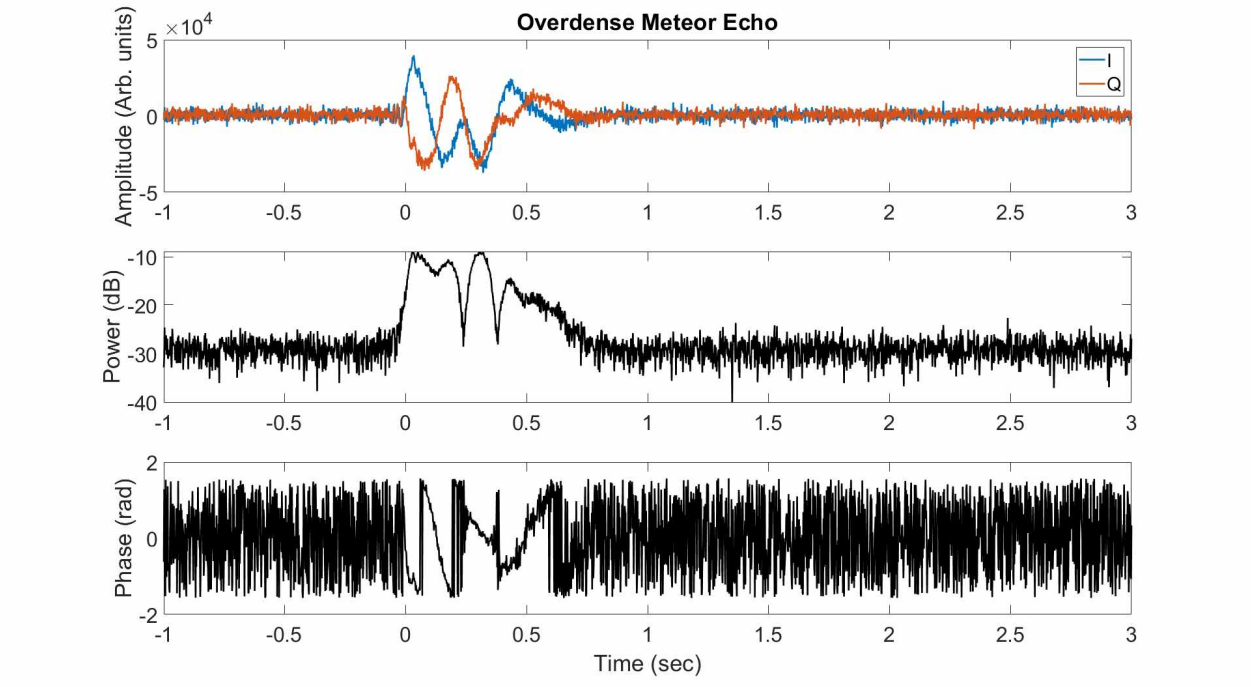

Figure 1.2: Example of an overdense meteor trail echo, exhibiting characteristic behavior in received power.

As previously stated, the transition between underdense and overdense meteor trail echoes is determined by the electron density of the meteor trail and the frequency of the observing radar. These parameters along with the initial radius of the meteor trail determine the plasma frequency of the trail. If this plasma frequency is greater than the radar frequency, the trail will appear overdense. An expression for the plasma frequency at its peak electron density is given in *Hocking et al.* [2016] and shown below in Equations 1.1 and 1.2.

$$f_p = \frac{e_0^-}{2\pi r_0 \sqrt{N_l / \pi \epsilon_0 m_e}} \tag{1.1}$$

$$N_l = \pi a^2 N_0 \tag{1.2}$$

In these equations, e_0^- is the electronic charge, r_0 is the initial trail radius, ϵ_0 is the permittivity of free space, N_l is the electron line density, and a is the radius at which the volume density falls to $1/e$ (where e is Euler's number) of the peak volume density, N_0. The signal-to-noise ratio (SNR) of the received signal power from an underdense echo is shown in Equation 1.3 [*McKinley*, 1961]. D is the ambipolar diffusion coefficient, t is time from the peak received power, r_0 is the initial trail radius, and λ is the radar wavelength. This model equation shows the dependence of the signal decay on the ambipolar diffusion coefficient. Since the initial trail radius cannot usually be measured by radio observers, the second exponential term in Equation 1.3 is usually collapsed into a generic amplitude term, and the ambipolar diffusion coefficient becomes the only driver for underdense meteor trail decay.

$$SNR = e^{-(32\pi^2 Dt)} e^{-(8\pi^2 r_0^2 / \lambda^2)} \tag{1.3}$$

For both underdense and overdense echoes, several parameters of the meteor trail can be measured. The most common of these includes echo range, echo phase, echo power, and angle of arrival. While echo duration can be measured for both underdense and overdense echoes, the decay times measured from underdense echoes are of special interest. This underdense

meteor trail decay time is measured from the time of maximum amplitude to the time when the amplitude has reached some fraction of the peak. Sometimes it is useful to define this as half of the maximum amplitude, while other times it is more useful to use the 1/e decay time, where e is once again Euler's number. For most underdense meteor trail echoes, the 1/e time constant of the echo is on the order of 0.1 seconds [*McKinley*, 1961]. The only driver in trail decay for short-duration trails is ambipolar diffusion, which reduces the volume density of the trail until it is no longer visible to radio observers. For underdense meteor trail echoes, it is useful to measure the echo decay time as the time it takes for the received power to decay by a factor of e^{-1}. This decay time is related to the ambipolar diffusion coefficient through Equation 1.4, where T_{un} is the time for the received power to drop to $1/e$ of the maximum, λ is the radar wavelength, and D is the ambipolar diffusion coefficient [*McKinley*, 1961].

$$T_{un} = \frac{\lambda^2}{16\pi^2 D} \tag{1.4}$$

Longer lasting meteor echoes are more difficult to analyze in terms of decay time. This is because there are additional effects that influence the decay time on longer time scales such as turbulent mixing, plasma instabilities, and chemical effects, in addition to the ambipolar diffusion that dominates the short-lived underdense echoes. Understanding these effects will be incredibly important in the processing of overdense meteor echoes. Some of these chemical effects have been explored over the lifetime of meteor radar studies, and may be significant for long lasting meteor echoes. Ozone, in particular, can be a significant driver for decay of long lasting meteor trails.

The lifetime of ozone in the MLT region is shorter than atmospheric transport scales, making it a molecule of particular interest in atmospheric studies. In addition, ozone in the MLT region has wide ranging effects that include: the thermal structure and dynamics of the upper atmosphere, chemical evolution of the atmosphere on geologic timescales, and operation of low earth-orbit satellites *Cevolani and Pupillo* [2003]. Finally, there exists a

seasonal variation in ozone in the middle atmosphere that is likely due to gravity wave induced transport *Thomas et al.* [1984]. This makes ozone a useful indicator for gravity wave activity in the middle atmosphere. A comprehensive overview of meteoric chemistry in the MLT region is given in *Baggaley* [1979], *Jones et al.* [1990], and *Plane* [2003]. Only the chemistry most directly relating to ozone and meteor plasma trails is reviewed here. The dominant chemical reactions between meteoric ions and ozone are shown in equations 1.5-1.7 below, where M is a meteoric ion (usually Fe or Mg). Firstly, a meteoric ion combines with an ozone molecule resulting in a metal oxide ion and diatomic oxygen.

$$M^+ \quad + \quad O_3 \quad \rightarrow \quad MO^+ \quad + \quad O_2 \tag{1.5}$$

Secondly, there are two reactions that take place between the metal oxide ion, free electrons, and neutral oxygen atoms.

$$MO^+ \quad + \quad e^- \quad \rightarrow \quad M \quad + \quad O \tag{1.6}$$

$$MO^+ \quad + \quad O \quad \rightarrow \quad M^+ \quad + \quad O_2 \tag{1.7}$$

The reaction shown in Equation 1.6 proceeds much faster than the reaction in Equation 1.7, so it dominates and results in the reduction of electron density of the meteor trail. The motion of these ions is driven through two processes: thermodynamic forces arising from temperature gradients and Coulomb forces from imbalances in ion charges [*Jones and Jones* [1990]]. Although other reactions occur between meteoric ions and atmospheric particles that contribute to the de-ionization of the meteor trail, the reactions shown here are thought to dominate the de-ionization process.

The echo decay time for an long-duration meteor trail is dependent on four factors, as shown in *Baggaley* [1978]. These include: initial electron line density, ambipolar diffusion of meteoric plasma, diffusion by turbulent mixing, and chemical processes in the meteoric

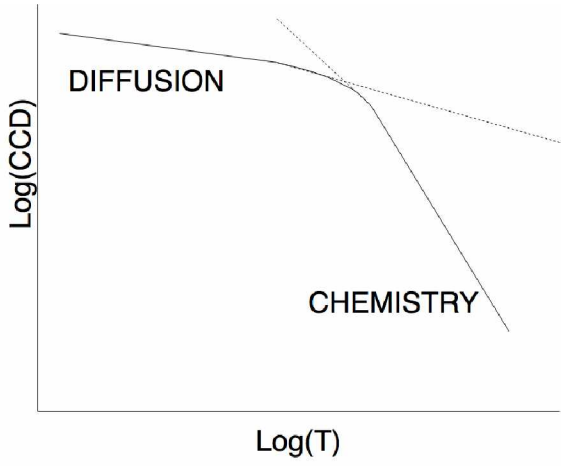

Figure 1.3: Adapted from *Jones et al.* [1990], a CCD of meteor echo decay times, illustrating the inflection formed by the diffusion and chemistry regions of trail decay.

plasma. There exists two major regions in the decay time for an long-duration meteor trail. For a short time after formation, the trail dissipation is dominated by ambipolar diffusion, before the chemical processes begin to take effect. After some initial time, the chemical de-ionization process becomes significant, speeding up the decay time for the meteor trail *Jones et al.* [1990]. A common tool in visualizing these two decay regions is the complementary cumulative distribution (CCD) of radio echo decay times. This is usually done by plotting the number of meteors with half-amplitude decay times greater than the time constant, T, versus the value of T itself. An example of this is shown in Figure 1.3, as seen in *Jones et al.* [1990]. In the absence of chemical reactions, the distribution of echo lifetimes should be a power-law reflecting the distribution of meteoroid masses *Ceplecha et al.* [1998]. This is because the lifetime of a meteor trail that is only influenced by ambipolar diffusion is related to the mass of the meteoroid, since meteoroids with more mass will create more ionization. The inflection in this plot corresponds to the critical value of the meteor echo duration, T_c. This critical time can be used to estimate ozone density in the mesosphere *Hocking et al.* [2016]. It is important to note that observed lifetimes are usually much longer for forward-scatter radar observations than for back-scatter. According to *Hocking et al.* [2016], it is rare to observe back-scatter meteor echoes that are longer than one second. Therefore, it

may be difficult to observe the inflection point in the CCD with back-scatter radar data. Decay times measured with back-scatter radar are used to illustrate some of these effects in Chapter 4.

Most current meteor radar processing depends on an assumption that underdense meteor backscatter is most significantly driven by ambipolar diffusion, and not any other effects. If that assumption breaks down, it is no longer possible to estimate ambipolar diffusion using current meteor radar techniques. Further work is needed to model meteor backscatter in greater detail, and determine under what conditions the ambipolar diffusion model of meteor decay is acceptable, and when other effects become too large to use that model reliably.

1.3 Thesis Overview

This thesis is organized into five chapters. Chapter 2 presents a summary of detection and classification of meteor echoes. This includes a detailed explanation of meteor radar interferometry used in typical modern meteor radar systems. Also presented is a comparison of methodologies for meteor classification, with discussion of the advantages and disadvantages of each approach. Finally in this chapter, an overview of several typical parameters that can be measured or calculated using meteor radar techniques is discussed.

Chapter 3 will include typical data products from Poker Flat Meteor Radar, including meteor counts, basic parameters, distribution of decay constants, and winds. Data shown in this chapter will span the period of November 2018 through August 2019. Standard data products for meteor radar systems usually include estimations for meteor position and velocity. The positions are usually calculated using phase differences on spaced receiver antennas, while the velocity can be determined either from the Doppler frequency of the meteor, or using the rate of change of the signal phase. Both the meteor position and velocity are used to calculate neutral winds in the meteor region. More information on how this information is calculated is included in this chapter.

Chapter 4 will cover the use of meteor observations to estimate neutral temperature, along with the current issues in that area and some possible explanations for why those estimates are so difficult to obtain. Also included in this chapter is data from simultaneous observations at Poker Flat Research Range with a meteor radar, a Rayleigh lidar, and a Sodium Resonance Wind Temperature Lidar (SRWTL). The meteor radar estimates the ambipolar diffusion coefficient, the Rayleigh lidar estimates neutral density, and the SRWTL estimates neutral temperature. There is a model equation that relates these three parameters, so if two are known then it should be possible to calculate the third. Difficulties in performing these calculations are explored in this chapter. Chapter 5 will cover conclusions of the work done in this thesis as well as future work that is needed in this area. Further discussion of current issues in meteor radar processing is also included.

Chapter 2: Meteor Detection and Classification

2.1 Introduction

A typical goal of meteor radar systems is to detect underdense meteor echoes and determine the following parameters: location of the meteor trail (usually in the form of a height, zenith angle, and azimuth angle), Doppler velocity, and decay time of the meteor trail. These data are then used to estimate atmospheric winds and temperature in the MLT region. Processing meteor radar data starts with determining the location of the radar echo, classifying the radar echo (i.e. sorting into underdense, overdense, others), then applying the appropriate algorithm to determine the echo velocity and decay time constant. There are several methods of processing the radar data to determine these parameters.

A popular method for determining the angle of arrival uses 5 antennas, configured in 4 pairs as shown in Figure 2.1. With this configuration, the phase differences between antenna pairs can be used to determine the angle of arrival [*Jones et al.*, 1998]. Many classification algorithms have been developed to determine whether an echo is from an underdense meteor or not. After the position of the meteor has been determined, the signal from the five receiver antennas can be coherently integrated to improve the signal to noise ratio before further processing. From this point, underdense meteor echoes are usually identified using a set of criteria to reject echoes that do not fit the signal model that underdense meteor echoes are expected to follow, as in *Holdsworth et al.* [2004]. Once an echo has been determined to be an underdense meteor, standard parameters such as meteor position and radial velocity are calculated, as in *Hocking et al.* [2001].

There are several meteor radar systems that have become somewhat popular for this type of work. One of these is the SKiYMET (All-Sky Interferometric Meteor Radar) system, a joint development of Genesis Software and MARDOC Inc. Genesis provides software with the SKiYMET system to allow for easy configuration and immediate data collection after the instrument has been installed. The standard data product of the Genesis software is an

MPD (Meteor Position Data) file that contains all of the radar detections for one day, along with calculated parameters such as height, zenith and azimuth angles, radial velocity, and decay time. Later in this chapter, the results of the Genesis software will be compared with other meteor radar data processing methods.

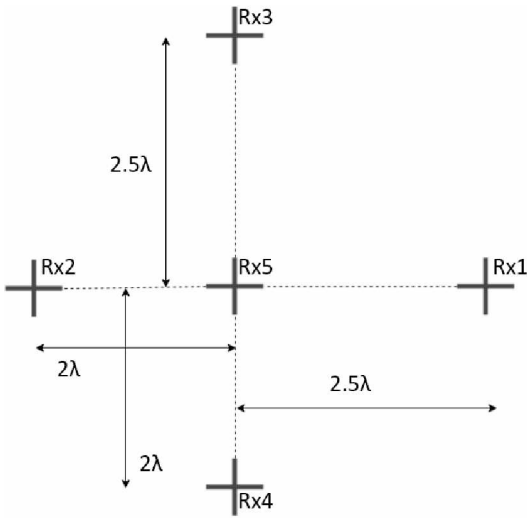

Figure 2.1: Meteor radar interferometer in Jones configuration.

2.2 Radar Interferometry

An general overview of radar interferometry as it applies to meteor studies is given in this section, and is summarized from *Jones et al.* [1998]. Figure 2.2 shows a set of linearly spaced antennas and an incoming signal with an angle of arrival, ϵ.

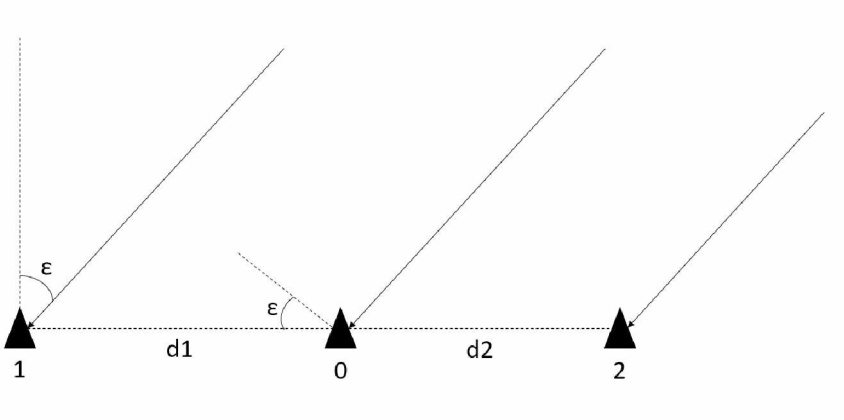

Figure 2.2: A diagram showing the measurement of angle of arrival, ϵ, using the measured signal phase on spaced antennas.

The angle of arrival (AOA) of an incoming radio wave can be determined using the phase angle difference, ϕ_{10}, on a set of antennas with a spacing of d. The phase of the received signal on antenna 1 relative to antenna 0 is given below in Equation 2.1, where d_1 is the distance between the antennas and λ is the radar wavelength.

$$\phi_{10} = -2\pi \frac{d_1}{\lambda} \sin(\epsilon) \tag{2.1}$$

For a given antenna spacing, d, and error in phase measurement, $\Delta\phi_{10}$, the error in AOA, $\Delta\epsilon$, is given in Equation 2.2. For example, for a spacing of 0.5λ, an AOA estimate of 60 degrees, and a measurement error of 1 degree, the AOA error $\Delta\epsilon$ would be 0.64 degrees. A spacing of 2.0λ under the same conditions would produce an error of 0.16 degrees, and a spacing of 2.5λ would have an AOA error of 0.13 degrees. It can be seen that as the antenna spacing increases, the error in the AOA estimate decreases. However, because phase angles between antennas are calculated in the range $[-\pi, \pi]$, only antenna spacings of 0.5λ or less can unambiguously determine the AOA. As the antenna spacing increases, it cannot be determined how many multiples of 2π have been introduced into the phase angle measurements, and those AOA estimates become ambiguous, although they have lower error than the shorter antenna spacings. One solution to this is to have multiple antenna spacings

so that an estimate with high error and no ambiguity along with a low error ambiguous estimate can be calculated and both can be used to calculate the true AOA.

$$\Delta\epsilon \approx \frac{\lambda\Delta\phi_{10}}{2\pi d\cos(\epsilon)} \tag{2.2}$$

For three spaced antennas as in Figure 2.2, the phase angle difference on the third antenna is calculated similarly to the one in Equation 2.1, and is given below in Equation 2.3.

$$\phi_{20} = +2\pi\frac{d_2}{\lambda}\sin(\epsilon) \tag{2.3}$$

For those three spaced antennas, there are two AOA estimates that can be calculated. These are shown in Equations 2.4 and 2.5. By adding or subtracting the phase angle difference across antenna pairs, AOA estimates for longer or shorter antenna spacings can be determined. This results in two AOA estimates as described above.

$$\sin(\epsilon) = -\frac{\lambda}{2\pi}\frac{(\phi_{10} - \phi_{20})}{d_1 + d_2} \tag{2.4}$$

$$\sin(\epsilon) = -\frac{\lambda}{2\pi}\frac{(\phi_{10} + \phi_{20})}{d_1 - d_2} \tag{2.5}$$

For example, if there are three antennas arranged as in Figure 2.2, and $\phi_{10} = +141.5°$, $\phi_{20} = +102.8°$, $\epsilon = -38°$, $d_1 = 2.5\lambda$, and $d_2 = 2.0\lambda$, then AOA estimates for both a 0.5λ spacing and a 4.5λ spacing can be calculated. The phase angle differences across antenna pairs will be $(\phi_{10} - \phi_{20}) = +38.7°$ and $(\phi_{10} + \phi_{20}) = -115.7°$. The AOA estimates can be calculated as shown in Equations 2.7 and 2.6.

$$\sin(\epsilon) = \frac{-(\phi_{10} + \phi_{20})}{\pi} \tag{2.6}$$

$$\sin(\epsilon) = \frac{-(\phi_{10} - \phi_{20}) \pm 2\pi n}{9\pi} \quad n = 0, 1, 2, 3, 4 \tag{2.7}$$

Because an arcsin operation is needed to calculate the AOA in equations 2.6 and 2.7, only values of $\sin(\epsilon)$ in the range of $[-1,1]$ are valid. For equation 2.6, this leads to an ambiguity in the AOA estimation giving nine possible solutions $[-52.3°, -39.6°, -26.8°, -14.1°, -1.4°, 11.3°, 24.1°, 36.8°, 49.6°]\pm 0.04°$ (assuming the phase measurement error is 1°). This is shown graphically in Figure 2.3. For equation 2.7 there is only one value in this range, $-40.0°\pm 0.4°$. Equations 2.6 and 2.7 provide ambiguous estimates with low errors and unambiguous estimates with high errors. As seen in Figure 2.3, only one estimate from equation 2.6 coincides with the estimate from equation 2.7. For this example, the calculated AOA would be $-38.2°$, since it is closest to the unambiguous estimate of $-40°$.

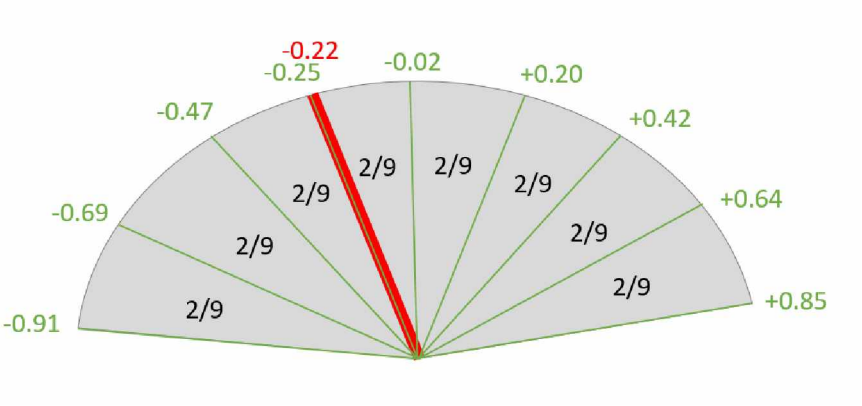

Figure 2.3: Large antenna spacings create ambiguity in the AOA estimate. For a spacings of 2.0λ and 2.5λ, as in the example discussed in this section, there are 9 possible AOA values. The array of AOA estimates calculated using Equation 2.7 is shown in green corresponding to the values in the $[-1,1]$ range. The AOA estimate from Equation 2.6 is shown in red. The thicknesses of the lines are indicative of their uncertainties.

This thesis will compare two methods of meteor radar interferometry. The first of these, *Jones et al.* [1998], has been in regular use since the late 1990's and the corresponding antenna configuration, shown in Figure 2.1, has become the standard configuration for most modern meteor radar systems. The second of these methods, *Vaudrin et al.* [2018], has been developed fairly recently, and claims to offer a more robust solution to meteor radar interferometry, as well as straightforward calculation of statistical uncertainty, which is lacking from the standard Jones method.

2.2.1 Classical Meteor Radar Interferometry

The method for meteor radar interferometry presented in *Jones et al.* [1998] utilizes phase differences between antennas that are spatially separated on two orthogonal baselines in order to estimate the angle of arrival of the radio echo and locate the meteor trail. Assuming an antenna configuration like that in Figure 2.1, there is a pair of antennas at 2λ and 2.5λ separations along each baseline. As stated in *Jones et al.* [1998], for antenna spacings larger than 0.5λ, there is ambiguity in the AOA estimation due to phase wrapping. However, for antenna separation that approaches 0.5λ, accuracy of AOA estimates is reduced. Mutual coupling between antennas further reduces accuracy for smaller antenna separations, so there will be no ambiguity and larger error in the estimate. Conversely, larger antenna separations will have lower error in AOA estimates, but will have more phase ambiguity as the distance between antennas increases. A technique often used to reduce both ambiguity and error is presented in *Jones et al.* [1998]. Using antenna spacings that differ by 0.5λ, as in Figure 2.1, allows for two AOA estimates that can be used together to get one unambiguous AOA estimate.

The phase difference on antenna 1 with respect to antenna 5 is given in *Jones et al.* [1998] as Equation 2.8, where d is the distance between antennas and ϵ is the AOA along that antenna baseline.

$$\phi_{15} = -2\pi \frac{d}{\lambda} sin(\epsilon) \tag{2.8}$$

The method for AOA estimation given in *Jones et al.* [1998] uses the phase differences of two antenna pairs (Equation 2.9), one at 2λ and one at 2.5λ, to get an estimate of the phase difference for an antenna pair of 0.5λ spacing. As stated earlier, this pair will have no ambiguity due to phase wrapping, but high error in the AOA estimate due to mutual coupling between antennas.

$$\phi'_{0.5\lambda} = \phi_{2.5\lambda} - \phi_{2.0\lambda} \tag{2.9}$$

Similarly, the two phase differences across these antenna pairs are summed to get an estimate of the phase difference across an antenna pair at 4.5λ spacing (Equation 2.10). The AOA estimate calculated from this antenna pair will have low error, but will have high ambiguity due to phase wrapping.

$$\phi'_{4.5\lambda} = \phi_{2.5\lambda} + \phi_{2.0\lambda} \tag{2.10}$$

To achieve a single AOA estimate for an antenna baseline using this method, the AOA estimate from the 4.5λ ensemble of estimates that is closest to the 0.5λ estimate is chosen. This is done for each baseline to get AOA estimates for each. Treating each of these AOA estimates as a direction cosine for a three-dimensional vector allows for the calculation of the direction cosine in the vertical dimension, using Equation 2.11.

$$\theta_z = \sqrt{1 - \theta_x^2 - \theta_y^2} \tag{2.11}$$

To calculate zenith (θ) and azimuth (Φ) angles, Equations 2.12a and 2.12b can be used after the direction cosines have been calculated. The direction cosines, θ_x and θ_y, are the AOA estimates from the two orthogonal antenna baselines. The phase differences across antenna pairs are related to the zenith and azimuth angles through Equation 2.13. Solving this system of equations is equivalent to determining the AOA for each baseline based on using a 0.5λ estimate and a 4.5λ estimate and calculating zenith and azimuth angles using Equations 2.12a and 2.12b. Equation 2.13 is shown to illustrate the relationship between measured phase differences across antennas and zenith and azimuth angles. In practice, the AOA estimate for each orthogonal baseline is calculated and the zenith and azimuth angles are calculated directly from those.

$$\theta = \arccos(\theta_z) \tag{2.12a}$$

$$\Phi = \arctan(\frac{\theta_y}{\theta_x}) \tag{2.12b}$$

$$\begin{bmatrix} \phi_{51} \\ \phi_{25} \\ \phi_{21} \\ \phi_{53} \\ \phi_{45} \\ \phi_{43} \end{bmatrix} = \frac{2\pi}{\lambda} \begin{bmatrix} 2\lambda & 0 \\ 2.5\lambda & 0 \\ 4.5\lambda & 0 \\ 0 & 2\lambda \\ 0 & 2.5\lambda \\ 0 & 4.5\lambda \end{bmatrix} \begin{bmatrix} sin(\theta)cos(\Phi) \\ sin(\theta)sin(\Phi) \end{bmatrix} \tag{2.13}$$

2.2.2 Complex Plane Interferometry

A method for parameter estimation using complex valued voltage signals from each antenna is presented in *Vaudrin et al.* [2018]. This method uses a non-linear least squares fitting technique to determine the parameters of an exponentially decaying sinusoid that best fits the underdense meteor trail echo. A model for the complex valued voltage signal at the output of a radar receiver is given in Equation 2.14. In this model, m is the sample number (corresponds to time), and k is the antenna number. The parameters calculated for a particular underdense meteor echo are as follows: amplitude on each antenna channel (A_k), Doppler frequency (f_d), diffusion coefficient (D), direction cosines along each antenna baseline (θ_x,θ_y), and a term that is the wavenumber multiplied by the range (ϕ_5). The angular wavenumber ($\frac{2\pi}{\lambda}$) is κ, and d_k and γ_k correspond to the distances and angular positions of each antenna with respect to a reference antenna.

$$V_q(m,k) = A_k e^{j(-\kappa d_k(\theta_x \cos\gamma_k - \theta_y \cos\gamma_k))} e^{j2\pi f_d t[m]} e^{-D_a t[m]} e^{j\phi_5} \tag{2.14}$$

Working the problem of meteor radar interferometry and parameter estimation in terms of complex valued voltage signals makes sense because the real and imaginary parts are already formed at the output of a pulsed Doppler receiver system as the in-phase and quadrature channels. The signals received on each antenna channel corresponds to the real or imaginary parts of the model signal in Equation 2.14. The model equation should be rewritten so that the signals on the in-phase and quadrature channels are expressed as real-valued functions, as in Equation 2.15. The vector containing the trial model parameters is β'.

$$\mathbf{\Omega}(\beta', m) = \begin{bmatrix} \Re(V_q(m,1)) \\ \Im(V_q(m,1)) \\ \Re(V_q(m,2)) \\ \Im(V_q(m,2)) \\ \Re(V_q(m,3)) \\ \Im(V_q(m,3)) \\ \Re(V_q(m,4)) \\ \Im(V_q(m,4)) \\ \Re(V_q(m,5)) \\ \Im(V_q(m,5)) \end{bmatrix}^T \tag{2.15}$$

Defining a residual matrix, $\mathbf{\Gamma}$, provides a good metric for how closely the model parameters are to the actual signal parameters. The residual matrix is defined in 2.16. $Y(m,k)$ is the measured signal on each receiver antenna.

$$\mathbf{\Gamma}(\beta', m) = \begin{bmatrix} f_1(\beta', m) \\ f_2(\beta', m) \\ f_3(\beta', m) \\ f_4(\beta', m) \\ f_5(\beta', m) \\ f_6(\beta', m) \\ f_7(\beta', m) \\ f_8(\beta', m) \\ f_9(\beta', m) \\ f_{10}(\beta', m) \end{bmatrix}^T = \begin{bmatrix} \Re(V_q(m,1)) - \Re(Y(m,1)) \\ \Im(V_q(m,1)) - \Im(Y(m,1)) \\ \Re(V_q(m,2)) - \Re(Y(m,2)) \\ \Im(V_q(m,2)) - \Im(Y(m,2)) \\ \Re(V_q(m,3)) - \Re(Y(m,3)) \\ \Im(V_q(m,3)) - \Im(Y(m,3)) \\ \Re(V_q(m,4)) - \Re(Y(m,4)) \\ \Im(V_q(m,4)) - \Im(Y(m,4)) \\ \Re(V_q(m,5)) - \Re(Y(m,5)) \\ \Im(V_q(m,5)) - \Im(Y(m,5)) \end{bmatrix}^T \mathbf{W} \tag{2.16}$$

$\mathbf{W}$ is a diagonal matrix of weights representing the inverse variances of the noise on each channel of the receiver antennas. The equation to be minimized is given in Equation 2.17.

$$\min_{\beta'} \sum |\mathbf{\Gamma}(\beta', m)|^2 \tag{2.17}$$

The use of a nonlinear least squares minimization algorithm, such as the Levenberg-Marquardt algorithm, will allow for estimates of the model parameters that are close to the actual signal parameters. In the case of no signal noise, the parameter estimates and the actual parameters will be exactly identical, however, there is always some level of electrical noise, meaning the minimum of Equation 2.17 will not be exactly zero. The Levenberg-Marquardt algorithm is described in Appendix A.

Due to the antenna spacing, there is significant ambiguity in the AOA estimates that are produced by standard methods. This ambiguity is illustrated in Figure 2.4 which was produced from a simulated meteor that was created using Equation 2.14. All of the parameters except for the AOA direction cosines were held constant, and the direction cosines were varied over an interval of [−1,1]. The objective function was inverted and plotted (since peaks are easier to spot than valleys). Because the Levenberg-Marquardt fitting algorithm

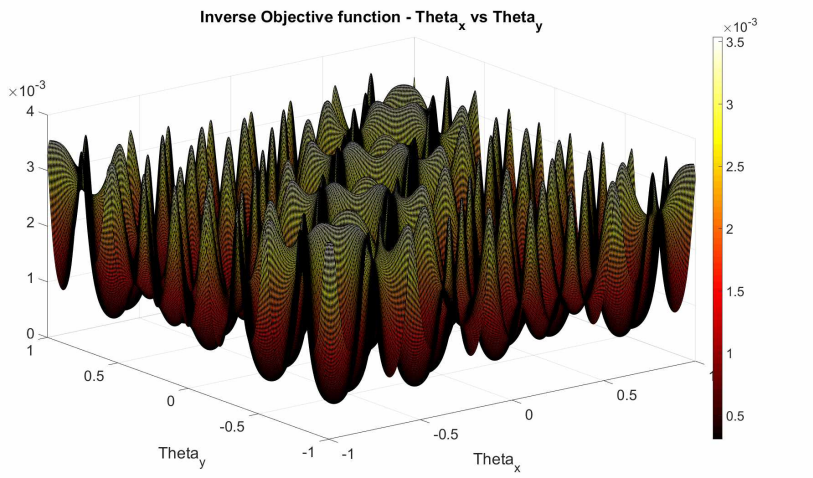

Figure 2.4: This figure was created by using the model in Equation 2.14, and varying the direction cosines while holding everything else constant. Because of the antenna spacing, there is significant ambiguity in the phase measurements. This creates many local minima in the objective function that can cause incorrect parameter values to be chosen by the nonlinear least squares minimization algorithm.

needs a starting point, and there are many local minima in the objective function, the Jones interferometry techniques were used to seed the fitting algorithm. Assuming these estimates are accurate, the fitting algorithm should slightly adjust the estimates and also allow for estimation of statistical uncertainty for fitted parameters. In addition, the amplitudes of each antenna channel are normalized to the maximum amplitude, so that all of the amplitudes are 1, allowing for an easy way to seed those parameters in the algorithm.

Another advantage to the complex interferometry method is that it allows for meteor radar measurement precision to be quantified in a way that is not common in modern meteor radar literature *Vaudrin et al.* [2018]. Because the parameter estimates from the nonlinear least squares fitting algorithm are an estimate of the true parameters, they have a statistical distribution that has some probability density. One way to determine the expected variation in any estimate of model parameters is to find the covariance matrix of the estimated parameters. The covariance matrix for the estimated parameters can be found by considering a matrix made up of the first derivatives of the residual matrix with respect to the model parameters, often called the "Jacobian." The Jacobian is defined in Equation 2.18 and is

calculated as a part of the Levenberg-Marquardt algorithm, so after the best fit parameters have been found, this information is already available. The covariance matrix, $\mathbf{\Sigma}$, is defined in *Vaudrin et al.* [2018] as in Equation 2.19, where $\epsilon = \sum \mathbf{\Gamma}(\beta', m)$, a is the number of simultaneous equations, and b is the number of parameters. Statistical uncertainties for the fitted parameters can be found using this method, but it is important to keep in mind that these are uncertainties for the fitted model parameters, not for any physical measurement.

$$\mathbf{J}(\mathbf{\Gamma}(\beta', m)) = \begin{bmatrix} \begin{bmatrix} \frac{\partial f_1(m_1)}{\partial \beta'(1)} & \cdots & \frac{\partial f_1(m_1)}{\partial \beta'(10)} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1(m_M)}{\partial \beta'(1)} & \cdots & \frac{\partial f_1(m_M)}{\partial \beta'(10)} \end{bmatrix} \\ \begin{bmatrix} \frac{\partial f_{10}(m_1)}{\partial \beta'(1)} & \cdots & \frac{\partial f_{10}(m_1)}{\partial \beta'(10)} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{10}(m_M)}{\partial \beta'(1)} & \cdots & \frac{\partial f_{10}(m_M)}{\partial \beta'(10)} \end{bmatrix} \end{bmatrix} \tag{2.18}$$

$$\mathbf{\Sigma} = \frac{\epsilon(\mathbf{J}^T\mathbf{J})^{-1}}{a - b} \tag{2.19}$$

2.2.3 Comparison of Interferometry Methods

To test the accuracy of each of these methods, a set 2500 simulated meteors was generated. The simulated meteors had direction cosines along each antenna baseline between -1 and 1, Doppler frequencies between -10 and 10 Hz, ambipolar diffusion coefficients between 2 and 6 m^2/s, and heights between 70 and 110 km. The SNR of the simulated meteors is uniformly distributed between 5 and 10 dB. To prevent matrix scaling issues for the Levenberg-Marquardt algorithm, the amplitudes of the received signal were normalized so that they all had a maximum amplitude of 1. This kept all of the model parameters on similar orders of magnitude, and should provide much better model fits.

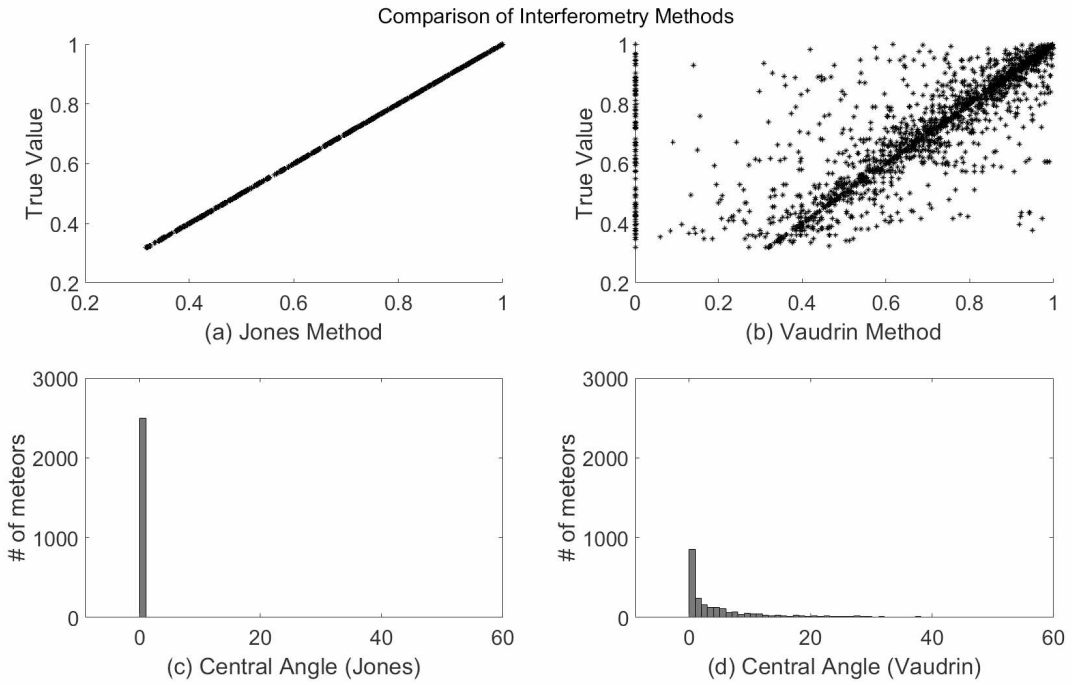

Figure 2.5: Simulated meteors were used to test the accuracy of interferometry methods. (a) The direction cosine in the vertical direction as calculated by the Jones interferometry method is plotted against the true values for the simulated meteors. (b) The direction cosine in the vertical direction as calculated by the complex interferometry method is plotted against the true values for the simulated meteors. (c) The errors in the AOA estimates are shown in a histogram for the Jones method. A value of 1 corresponds to a 100% error. (d) The errors in the AOA estimates are shown in a histogram for the complex interferometry method.

For Figures 2.5(a) and 2.5(b), the vertical direction cosine as calculated by each method was plotted against the true values of the direction cosine for the simulated meteors. In Figures 2.5(c) and 2.5(d), the central angle between the true meteor location and the estimated meteor location is calculated and plotted. Smaller central angles mean that the points are closer together. The central angle $\Delta\sigma$ is calculated as shown in Equation 2.20, where α_1 and α_2 are the elevation angles of the two points, ϕ_1 and ϕ_2 are the azimuth angles of the two points, and $\Delta\phi$ is the absolute difference in azimuth angle.

$$\Delta\sigma = \arccos(\sin(\alpha_1)\sin(\alpha_2) + \cos(\alpha_1)\cos(\alpha_2)\cos(\Delta\phi)) \tag{2.20}$$

The Jones method of interferometry seems to be incredibly robust in determining the correct AOA for underdense meteors. Even though the Vaudrin method is seeded with the Jones estimates, the nonlinear least squares fitting is not terribly stable, and it can sometimes choose one of the many local minima in the objective function that do not correspond to the actual angle of arrival. One advantage of the Vaudrin method for interometry is that it combines the interferometry and parameter estimation into one step. However, care should be taken to observe goodness of fit parameters for the Levenberg-Marquardt fit. For good fits, the error in AOA will be low, and the calculated parameters are likely to be correct. For received signals with worse fits, it may be better to use the Jones interferometry estimates.

2.3 Meteor Echo Classification Algorithms

Many classification algorithms for underdense meteor echoes focus on time domain processing, such as the algorithms presented in *Holdsworth et al.* [2004] and *Hocking et al.* [2001]. These algorithms tend to search for time domain characteristics that are typical of underdense meteor echoes. An echo that fits all the criteria is categorized as an underdense trail, while echoes that do not meet one or more criteria are rejected. Firstly, the signals on each of the five antennas are coherently combined using the phase differences calculated for each antenna. This greatly increases the signal-to-noise ratio and results in greater accuracy in classification of underdense meteors.

It is expected that an underdense meteor echo will have a rapid rise in signal amplitude followed by an exponential decay. One feature that makes an echo unlikely to be an underdense meteor echo is oscillations. These oscillatory echoes may be overdense meteor echoes, or some other types of echoes. Oscillatory echoes can be identified by looking for multiple peaks in received power over a short period of time.

The following sections will compare three classification techniques: the time series classification presented in *Holdsworth et al.* [2004], a support vector machine classification tech-

nique presented in *Zhao et al.* [2011], and the "Skicorr" classification technique presented in *Hocking et al.* [2001].

2.3.1 Time Series Classification

The most straightforward method for detecting underdense meteor echoes is to look for characteristics that are expected to appear in underdense meteor backscatter. If it is assumed that echoes from underdense meteors will follow the ambipolar diffusion model of meteor backscatter, there will be several features in the signal that can be searched for using an algorithm. Table 2.1 shows a set of criteria that can be used to reject echoes from objects that are not likely to be underdense meteors. These are based on the classification technique presented in *Holdsworth et al.* [2004]. Any echo that does not meet any of these criteria is likely to be from an underdense meteor. By removing echoes that are not underdense meteors, it is assumed that everything that is left is a true underdense meteor. This assumption is likely not always valid, as other types of echoes can possibly meet these criteria.

Table 2.1: Rejection criteria for time series classification.

	Rejection Criteria
1	Low SNR
2	Angle of arrival not feasible
3	Echo at start of end of time series
4	Echo less than 5 samples long
5	Power spike before echo
6	Power spike after echo
7	Oscillatory echo

Criterion 1 rejects signals that are very noisy, and are therefore either not likely to be underdense meteor backscatter, or are too noisy to process accurately anyway. Setting the SNR threshold for rejection to something somewhat low, such as around 4 dB, seems to be the best approach. Any signal with an SNR below that will not be processed accurately, and if the user of the data product wants to use only signals with higher SNR, it is straightforward to filter them out using the SNR field in the data file. The format of the Poker Flat Meteor

Radar level 1 data product is outlined in Appendix C. Figure 2.6 shows an example of a detection that is rejected by the classification algorithm due to low SNR. The SNR of this detection is below the threshold of 4 dB, so it is not classified as an underdense meteor. Criterion 2 rejects detections that cannot be accurately processed because the calculated angle of arrival is not feasible. If the calculated zenith angle is over 85 degrees, or if it is complex, the detection is rejected. A complex zenith angle occurs when the sum of the two direction cosines calculated from the antenna baselines is greater than 1. The vertical direction cosine is calculated from these direction cosines as shown in Equation 2.11.

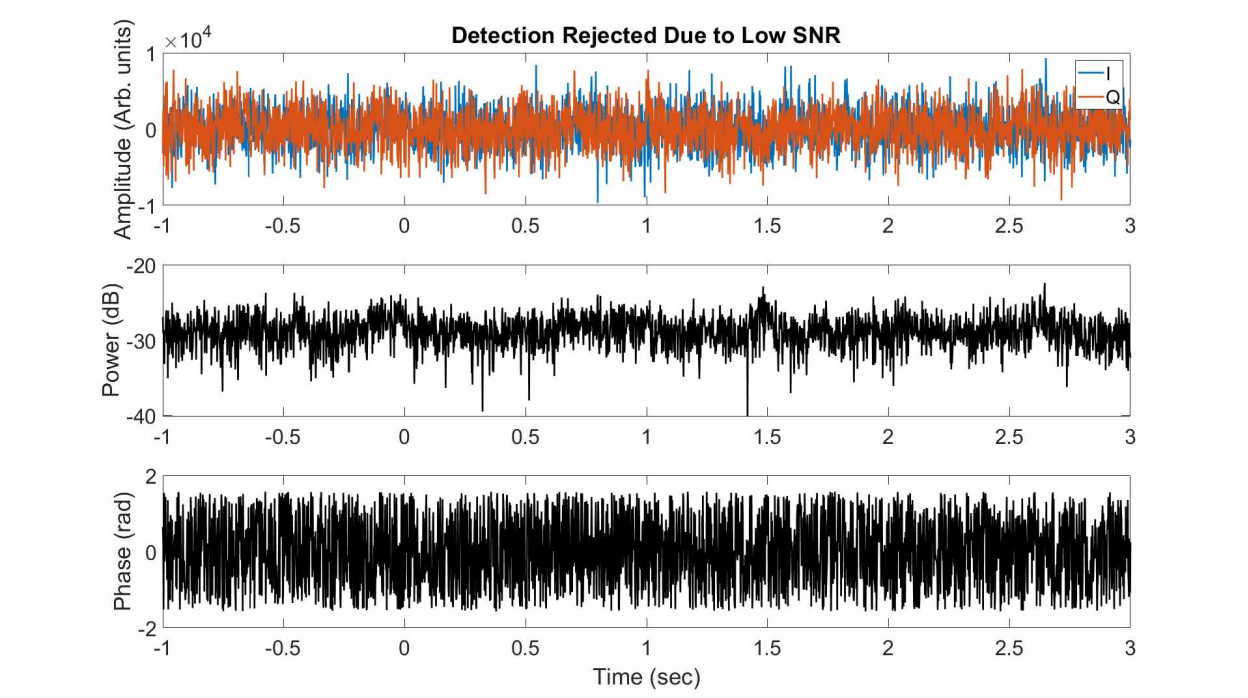

Figure 2.6: The SNR of this detection is below the threshold of 4 dB, so it is rejected by the classification algorithm.

Each raw data file saved by the radar contains 4 seconds of data, 1 second of data before the echo peak and 3 seconds after the peak. Criterion 3 rejects echoes that are not contained within a 4 second raw data file. If the echo does not decay back to the noise floor by the beginning or end of the raw data file, it most likely cannot be processed accurately. Figure 2.7 shows an example of one of these echoes.

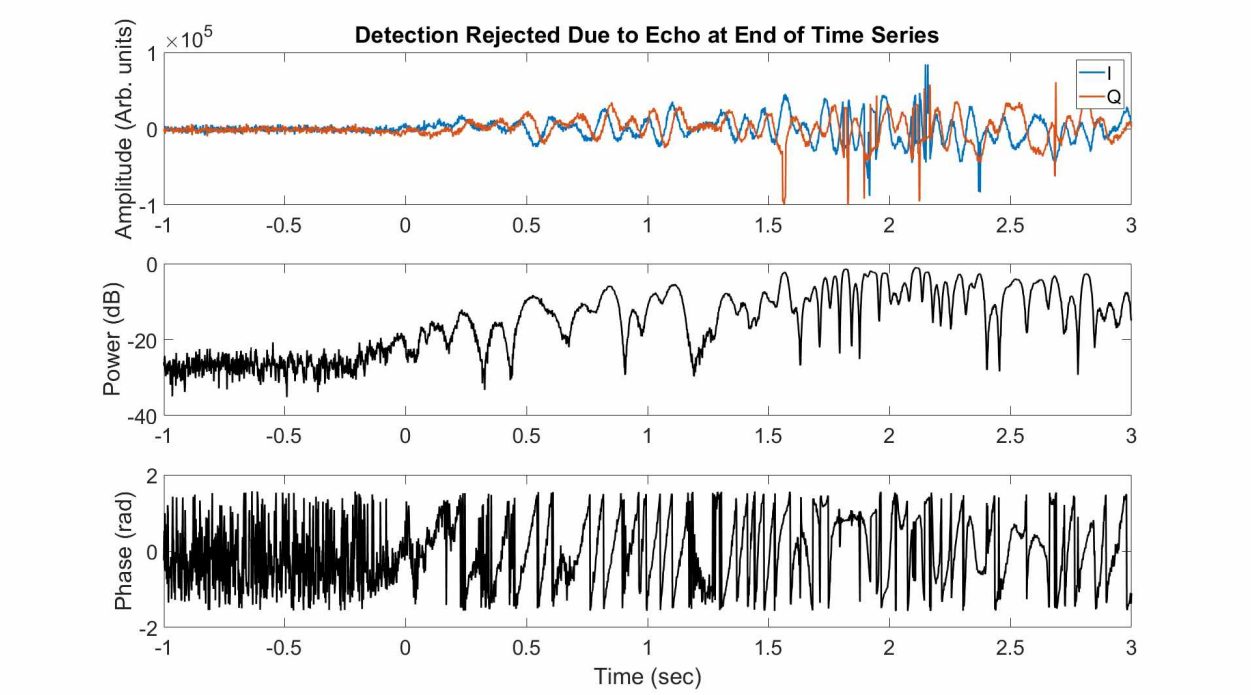

Figure 2.7: This echo does not decay to the noise floor by the end of the data file, so it is rejected by the classification algorithm.

Echoes that are less than 5 samples long are not likely to be accurately processed, so they are rejected by the classification algorithm. In addition, at most common meteor radar configurations 5 samples corresponds to less than 10 ms, and those short echoes are not likely to be useful even is they can be accurately processed as underdense meteors. Criterion 4 rejects these echoes. Figure 2.8 shows an example of a detection that was rejected due to being too short.

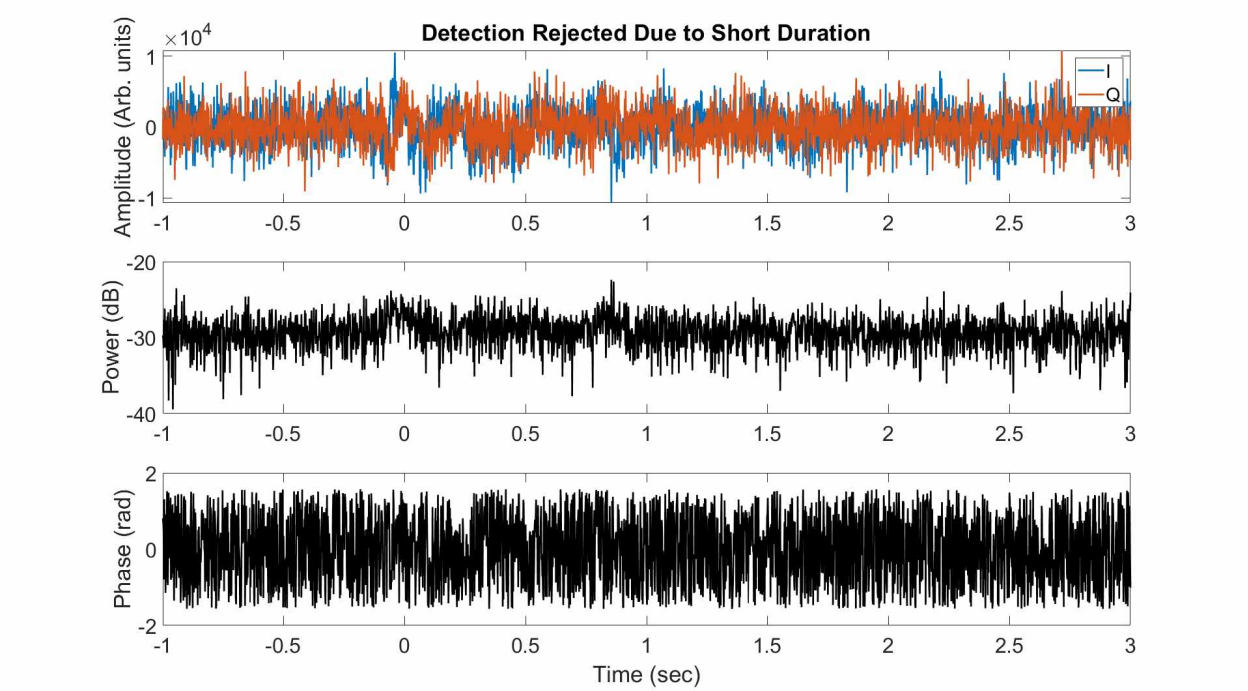

Figure 2.8: This echo is less than 5 samples long, so it is rejected by the classification algorithm.

Similarly to echoes that are not contained within the 4 second raw data file, echoes that have large spikes in received power before or after the echo has decayed back to the noise floor are considered to not be underdense meteors. This likely indicates that the entire echo has not been captured by the processing, or it is likely not an underdense meteor. Criteria 5 and 6 reject echoes that are of this type. These criteria are checked by normalizing the received power based on the peak received power, and if the average of 8 samples before or after the echo is above 0.5, the echo is rejected. Figure 2.9 shows an example of an echo that was rejected by the classification algorithm for having a spike in received power before the main echo. Figure 2.10 shows an example of an echo that was rejected by the classification algorithm for having a spike in received power after the main echo. Because of the large amount of noise in this echo, the end of the echo was determined to be around the 1 second mark in the figure. The small increase in received power after the end of the echo at around 1.5 caused this echo to be rejected. While it is possible to have multiple underdense echoes within a few seconds, it is much more common to have echoes from overdense meteors or

other sources that have large oscillations in received power. Some sections of these long duration echoes may appear to be underdense meteor echoes, so it makes sense to reject any echoes that occur closely in time to other echoes.

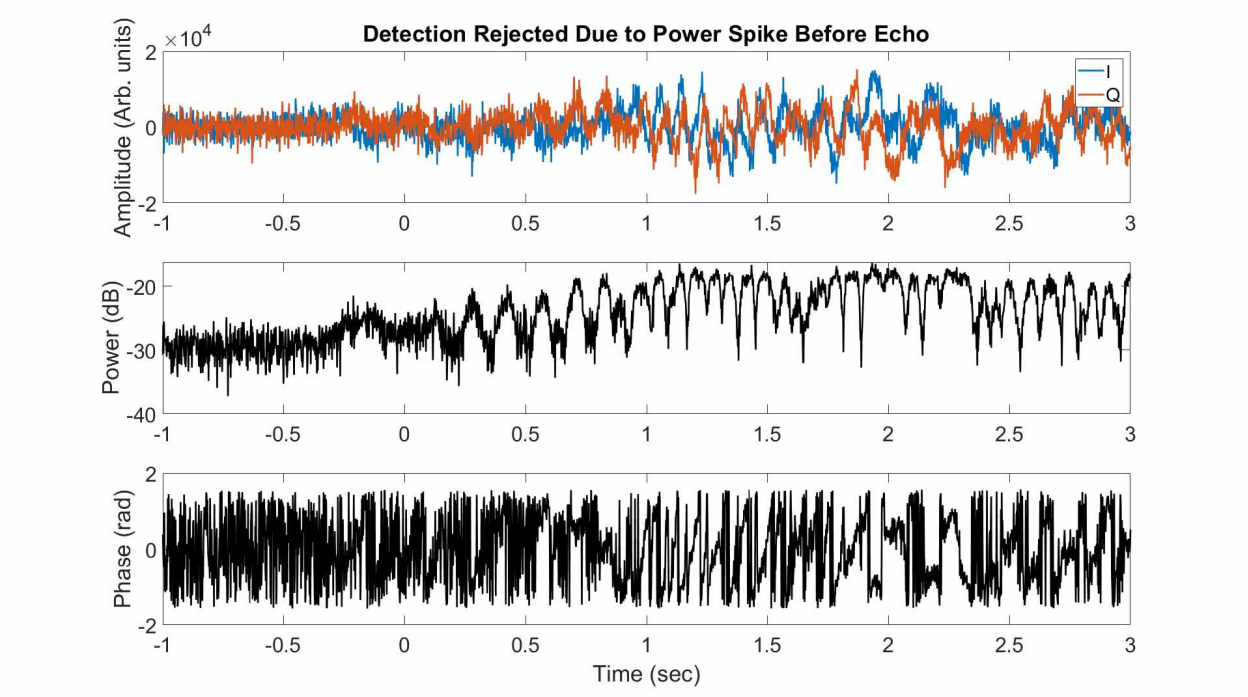

Figure 2.9: In addition to violating several other rejection criteria, this echo has a slight spike in received power just before the main echo (zero second mark in the plot).

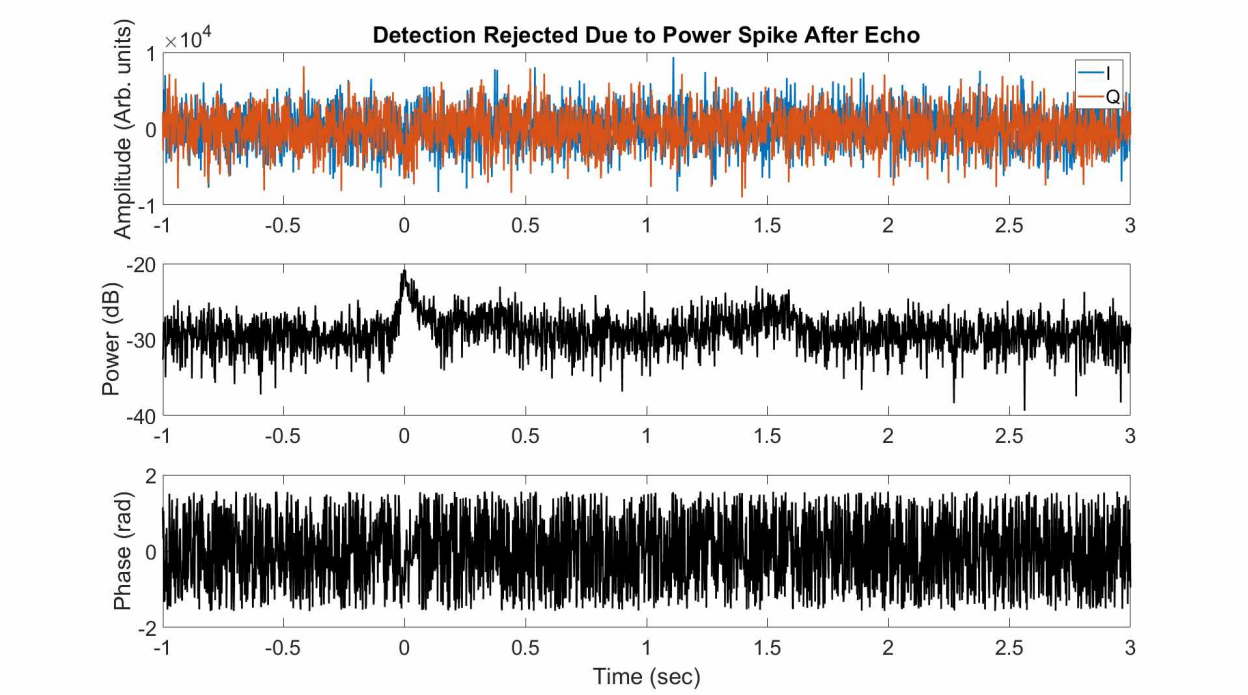

Figure 2.10: This echo has a spike in received power (around the 1.5 second mark in this plot) which is after the main echo, causing it to be rejected by the classification algorithm.

Echoes that have large oscillations in received power are not likely to be underdense meteors, but may be overdense meteors. Criterion 7 rejects echoes of this type, and allows them to be saved for later processing. Figure 2.11 shows an example of a detection that was rejected due to large oscillations in received power.

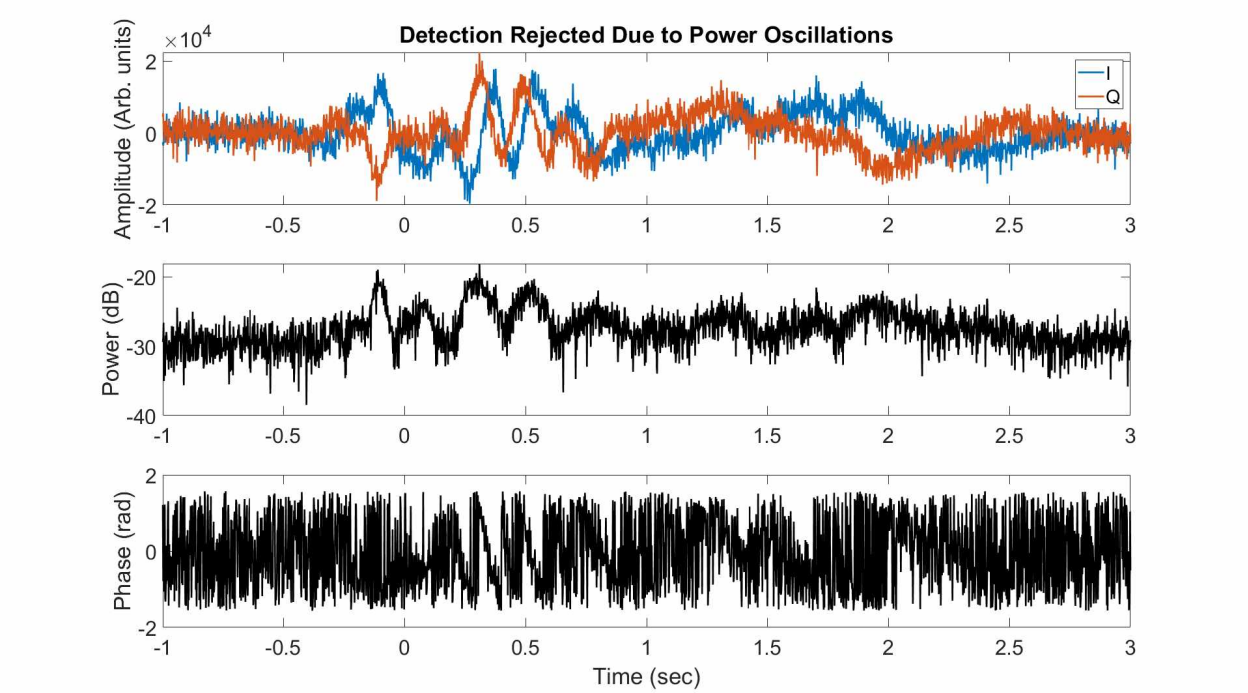

Figure 2.11: This echo was rejected by the classification algorithm for having large oscillations in received power.

To test the accuracy of this classification algorithm, a set of 307 radar detections were visually classified into one of three categories: underdense meteor, overdense meteor, or other. The underdense and overdense meteor categories were chosen to reflect typical characteristics as shown in Figures 1.1 and 1.2. Anything else was sorted into the other category. The results are shown in Table 2.2. The time series classification algorithm seems to perform pretty well for identifying underdense meteor echoes and rejecting non-meteor echoes. However, about 40% of overdense echoes were categorized as underdense meteors. This is likely due to small segments of overdense echo signals looking like underdense echo signals. It is possible for the classification algorithm to determine that a segment of an overdense echo is actually an underdense echo.

Table 2.2: Results from testing time series classification algorithm on a set of visually classified meteors.

Time Series Classification				
	Predicted			
Actual	Underdense	Overdense	Other	Total Objects
Underdense	91.58%	1.05%	7.37%	95
Overdense	41.94%	41.94%	16.12%	62
Other	4.67%	6.00%	89.33%	150

2.3.2 Support Vector Machine Classification

Support Vector Machines (SVMs) are a type of supervised machine learning that can be used to classify signals or images. The goal of a Support Vector Machine is to determine an optimal separating hyperplane which maximizes the margin of the training data. Figure 2.12 shows a 2-dimensional example of an SVM along with its training data, separating hyperplane, and margin. Although this example only has two dimensions, the same concept can be applied to to higher dimensions. In the figure there are two categories in the training data designated by color (red and blue). There are two hyperplanes that are drawn on the figure to show what an optimal SVM solution might look like. All of the red data and all of the blue data are completely separated by the hyperplanes. The margin is the distance between the two hyperplanes. To use the SVM for classification, this margin should be the largest possible. Training the SVM as a signal classifier involves maximizing the margin, M, and then classifying new data based on what side of the hyperplane it falls onto. The treatment of SVMs in this thesis is summarized from [*Kowalczyk*, 2017].

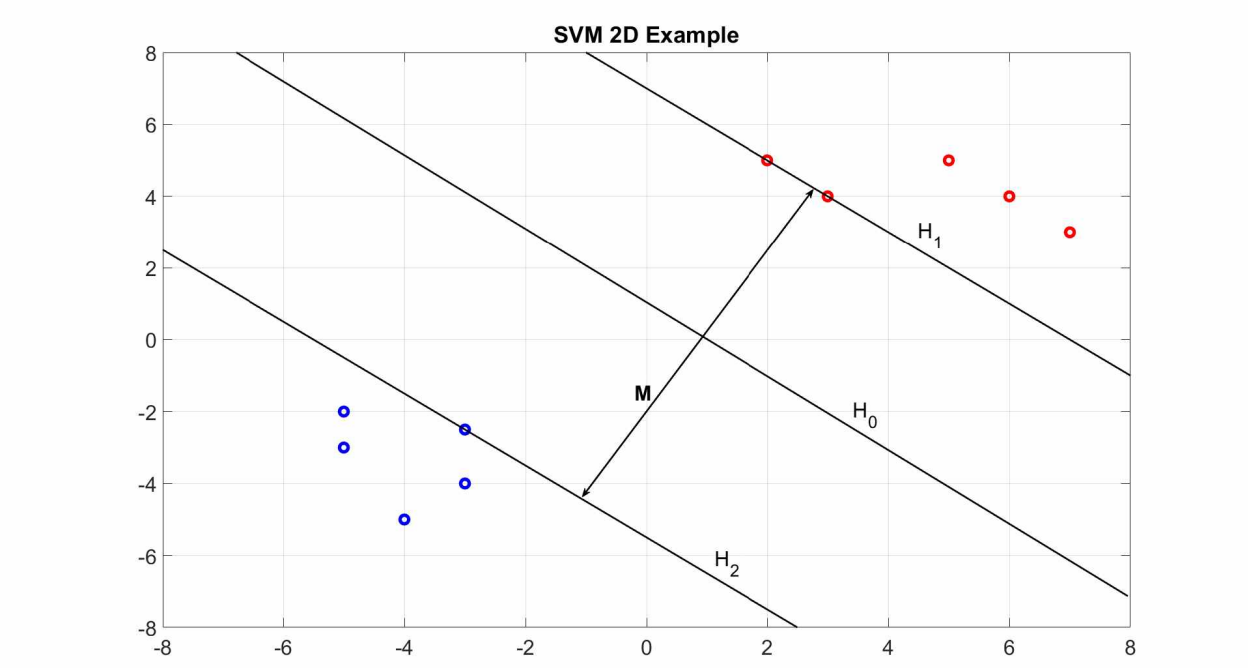

Figure 2.12: 2-dimensional example of an SVM classifier. Two groups of training data (red and blue) are separated by two hyperplanes with an optimally large margin, M.

The equation for a hyperplane is defined in Equation 2.21, where $\mathbf{w}$ is the vector of coefficients that define the hyperplane, and $\mathbf{x}$ is the vector of parameters that are being used to train the SVM.

$$\mathbf{w}^T\mathbf{x} = 0 \tag{2.21}$$

If one hyperplane is drawn between two data categories in the training data that is described by the equation in Equation 2.22, then two more hyperplanes can be drawn as well, described by Equation 2.23 and Equation 2.24.

$$\mathbf{H}_0 : \quad \mathbf{w}^T\mathbf{x} + b = 0 \tag{2.22}$$

$$\mathbf{H}_1 : \quad \mathbf{w}^T\mathbf{x} + b = +\delta \tag{2.23}$$

$$\mathbf{H}_2 : \quad \mathbf{w}^T\mathbf{x} + b = -\delta \tag{2.24}$$

Maximizing the distance between the hyperplanes optimizes the SVM classifier. There is a relationship shown in Equation 2.25 between the margin, M and the norm of $\mathbf{w}$. Minimizing $\|\mathbf{w}\|$ maximizes M.

$$M = \frac{2\delta}{\|\mathbf{w}\|} \tag{2.25}$$

Using a large set of training data, an SVM was trained to classify meteor echoes into three categories: underdense, overdense, and other. The MATLAB Machine Learning Toolbox was used to train two binary SVM classifiers. The classifiers used 15 parameters to make the determination between underdense meteor, overdense meteor, and other radar echoes. These parameters are listed in Table 2.3. The general flow of the SVM classification is shown in Figure 2.13. An echo is processed to determine the classification parameters, and then is fed through the underdense SVM classifier. This classifier will determine if an echo is "underdense" or "not underdense." The echoes that are not underdense are fed through an additional SVM classifier that is trained to look for overdense echoes. The result is three categories of echoes. Similarly to the time series classification, the SVM classification was tested on a set of visually classified echoes. The results are shown in Table 2.4. Only about half of the underdense meteors were correctly identified, and almost all of the "other" echo types were classified as overdense meteors. The time series classification seems to work much better, but there is still a lot of space to optimize the SVM classifier and tweak model parameters to likely increase performance.

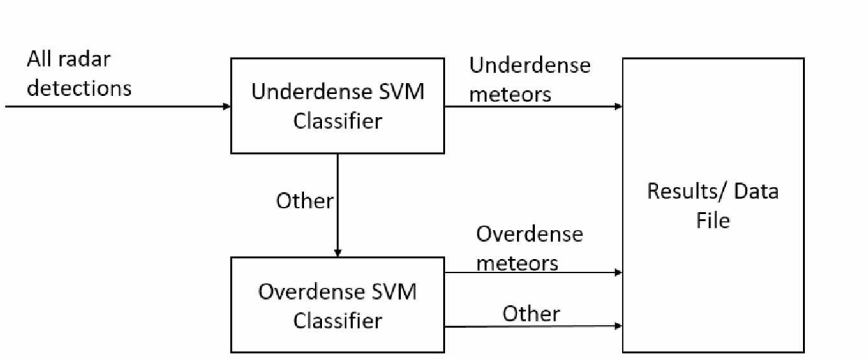

Figure 2.13: Flow chart showing how data moves through the SVM classification procedure. All radar detections are fed into the underdense SVM classifier, which sorts them into underdense meteors and other objects. These are fed into the overdense SVM classifier, which sorts them into overdense meteors and other objects.

Table 2.3: Parameters used for SVM classification.

SVM Parameters	
1	Duration
2	Maximum SNR
3-12	10 Evenly selected power data
13	Average of 8 power data before echo
14	Average of 8 power data after echo
15	Normalized error of model fitting

Table 2.4: Results from testing SVM classification on a set of visually classified meteors.

SVM Classification				
	Predicted			
Actual	Underdense	Overdense	Other	Total Objects
Underdense	52.63%	23.16%	24.21%	95
Overdense	24.19%	61.29%	14.52%	62
Other	0.00%	99.33%	0.67%	150

2.3.3 Skicorr Classification

The classification system in use in the Genesis software that comes with SKiYMET radar systems is based on the Skicorr classification scheme presented in *Hocking et al.* [2001]. There are three criteria that must be met under this classification method for an echo to be

categorized as an underdense meteor. Firstly, three data intervals of 0.25 seconds are isolated at the following locations: 1 second before the peak amplitude, 0.35-0.1 seconds before the peak amplitude, and 0.7 seconds after the peak amplitude. The peak must be at least twice the amplitude of the three data samples collected to be considered an underdense meteor. This criteria utilizes the fact that underdense meteors are usually short duration events and are unlikely to occur adjacent to one another in time. Secondly, 1 second of data before the peak amplitude is cross-correlated on different channels. There must be low correlation for the echo to be considered an underdense meteor. Thirdly, data from the duration of the echo is cross-correlated on different channels, and high correlation is indicative of an underdense meteor echo. The same set of visually classified meteors that was used to test the other classification methods was used for this method as well. All of the echoes were classified as underdense meteors. The three criteria used in this method do not seem to be very robust in rejecting radar echoes that do not conform to the ambipolar diffusion model of meteor trail decay.

Table 2.5: Results from Skicorr classification on a set of visually classified meteors.

Skicorr Classification				
	Predicted			
Actual	Underdense	Overdense	Other	Total Objects
Underdense	100.00%	0.00%	0.00%	95
Overdense	100.00%	0.00%	0.00%	62
Other	100.00%	0.00%	0.00%	150

2.4 Algorithms in use for Poker Flat Meteor Radar Processing

The typical processing algorithms that are used on Poker Flat Meteor Radar data were chosen based off of the analysis presented in this chapter. The time series classification scheme presented in section 2.3.1 is used for underdense meteor echo classification, since at the time of writing this thesis, it significantly outperforms the SVM classification. Echoes that are found to have large oscillations in received power are flagged accordingly for later processing as "possibly overdense." All echoes have AOA estimates using the Jones inter-

ferometry presented in section 2.2.1. For underdense echoes, this initial AOA estimate is used to seed the complex interferometry processing presented in section 2.2.2. Both sets of AOA estimates will be included in the level 1 data product described in Appendix B. The residual from the nonlinear least squares fit will also be included for every underdense meteor as a goodness of fit metric. This allows statistical uncertainties for fitted parameters to be included in the data file, while also allowing bad model fits to be easily identified.

Chapter 3: Poker Flat Meteor Radar Data

This chapter contains data from Poker Flat Meteor Radar (PFMR)(65.13°N,147.49°W). Spatial and temporal distributions of detections are shown, along with some preliminary data products. Poker Flat Meteor Radar is a standard SKiYMET installation and is configurable to a certain extent. The typical operating parameters are listed in Table 3.1. The radar was installed at Poker Flat Research Range in November 2018 and has been operating almost continuously since then. One of the receiver antennas is shown in Figure 3.1. There are five receiver antennas and one transmitter antenna arranged as shown in Figure 2.1. All six are identical crossed dipole antennas.

Table 3.1: Poker Flat Meteor Radar typical operating parameters.

Poker Flat Meteor Radar Typical Operating Parameters	
Transmit Frequency	32.55 MHz
Pulse Repetition Frequency	625 Hz
Inter-Pulse Period	1.6 ms
Pulse Type	7-bit Barker sequence
Transmit Power	30 kW

Figure 3.1: One of the receiver antennas that make up Poker Flat Meteor Radar.

3.1 Meteor Data From November 2018 Through August 2019

Figure 3.2 contains the number of objects detected by the radar every day. There is a seasonal variation that can be seen with a minimum near the spring equinox and maxima near the summer and winter solstices. This seasonal trend is expected and was also observed in *McKinley* [1961]. This seasonal variation is likely due to differences in meteor concentration in different parts of the Earth's orbit, as observed in *Mawrey and Broadhurst* [1993]. The radar was down for maintenance several times which corresponds to the days with low or no detection counts. There are large spikes in December and January that correspond to the Geminids and Quadrantids meteor showers, respectively. It is also likely that there are echoes included that do not belong to underdense or overdense meteors. Given the latitude of PFRR, the radar will likely see echoes from aurora as well as polar mesospheric summer echoes (PMSE). A portion of the larger number of detections in the summer may be due to PMSE.

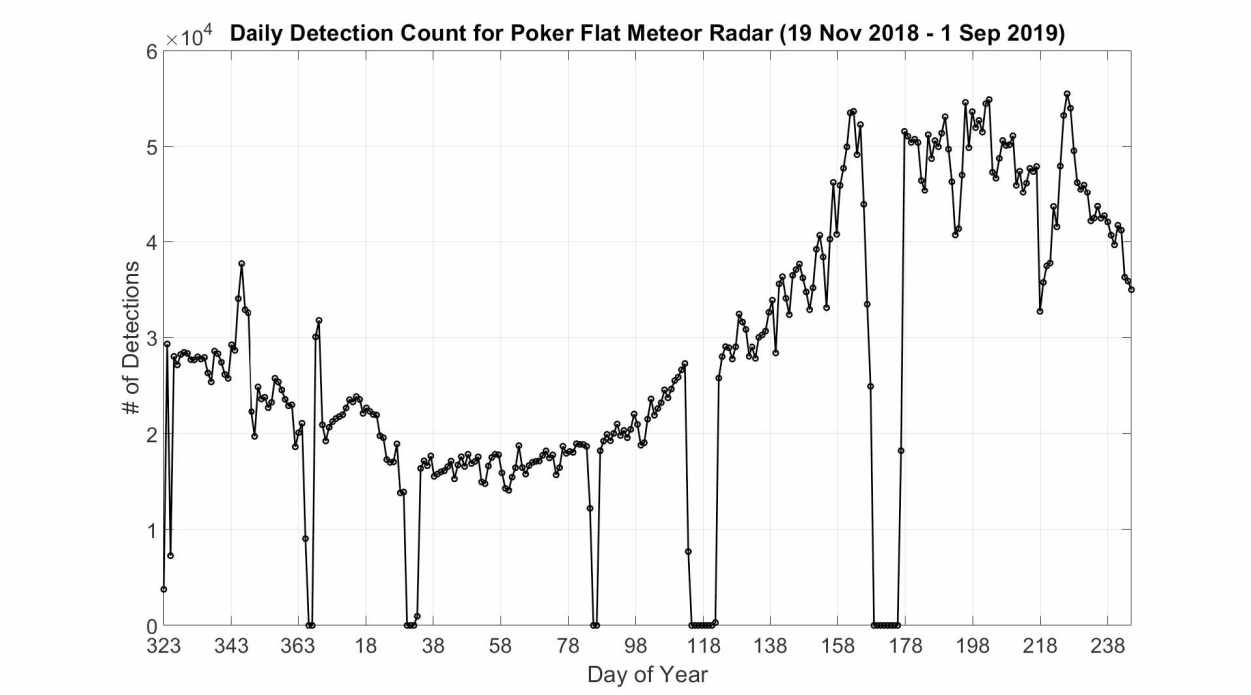

Figure 3.2: Number of detections for each day by Poker Flat Meteor Radar from Nov 19, 2018 through July 15, 2019. Days with low or zero counts are due to the radar being offline for maintenance.

Figure 3.3 shows the height distribution of all detections. Most objects are detected between 80-100 km, and the peak is at 90 km. For atoms on the surface of the meteor to start to ablate and form the meteor trail, a surface temperature of about 2200 K must be reached [*Ceplecha et al.*, 1998]. This usually occurs around 80-90 km since that is the height range where the atmosphere starts to get dense enough to cause friction heating. As the meteoroid continues to ablate it slows down as it loses mass. After traveling several kilometers ablation usually stops because the meteoroid has no more mass.

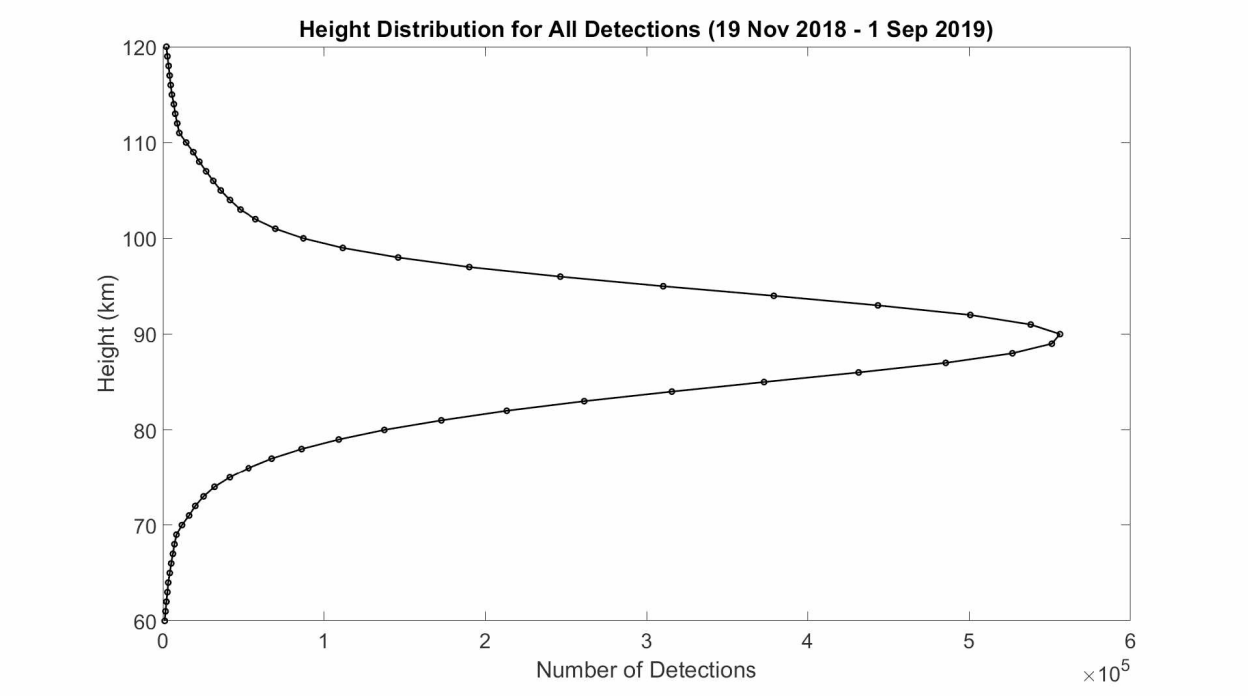

Figure 3.3: Height distribution for all detections from Nov 19, 2018 through July 15, 2019. The detections are binned at 1 km intervals. The peak is at 90 km.

Figure 3.4 shows the distribution of zenith angles for all detected objects. The bite-out (dip in number of detections) observed at approximately 70 degrees off of vertical is due to the PRF of 625 Hz that is in use for Poker Flat radar operation. The receiver is blanked when the transmitter is on. Decreasing the PRF would move the bite out closer to 90 degrees, however, there would also be less samples per meteor echo. The peak of the distribution is at about 60 degrees. The radar usually only detects specular reflections, meaning that a

meteor detected at 0 degrees would have to be parallel to the ground. This meteor would have traveled through a lot more atmosphere than a meteor detected at a higher zenith angle, so the meteoroid would have to be exceptionally large to continue ablating that long.

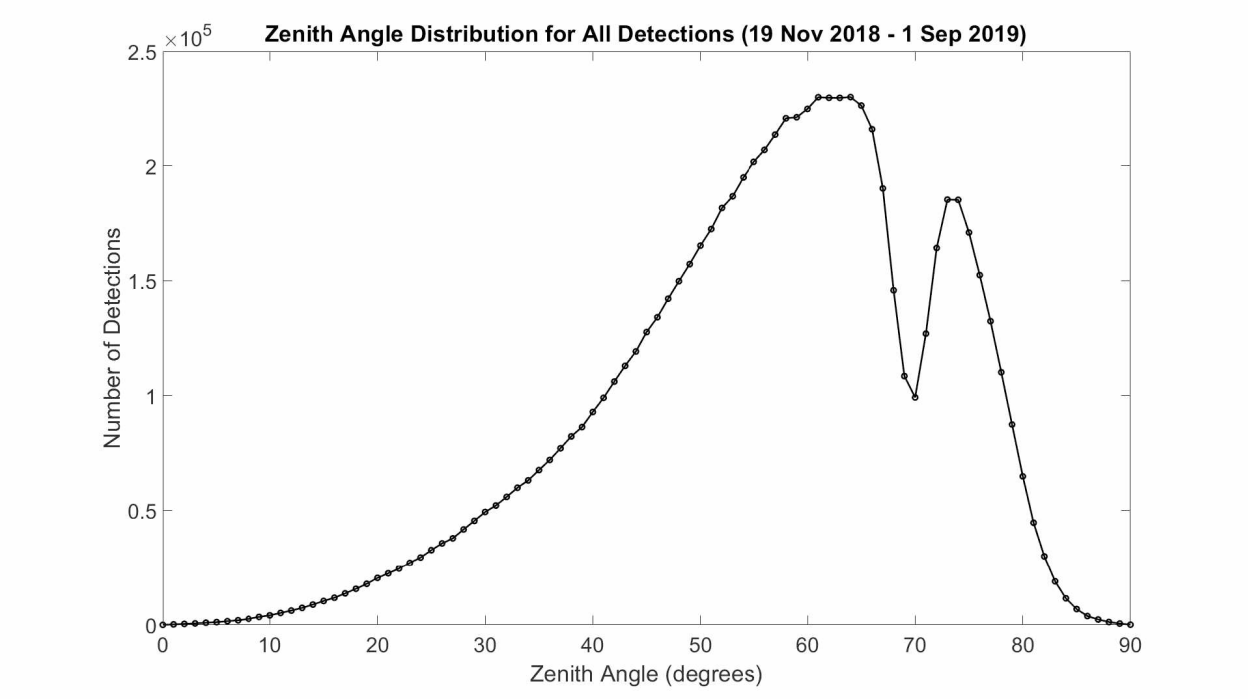

Figure 3.4: Zenith angle distribution for all detections from Nov 19, 2018 through July 15, 2019. The detections are binned at 1 degree intervals. There is a bite-out (dip in number of detections) at approximately 70 degrees due to the PRF that is being used.

Figure 3.5 contains the distribution of azimuth angles for all detected objects. There is a maximum around east (0 degrees) and a minimum around west (180 degrees). This is likely because east is the ram direction of the Earth as it rotates into space debris. It makes sense that more objects would be seen in this direction. Objects coming from the west would have to catch up, so only the highest velocity objects would be seen from that direction.

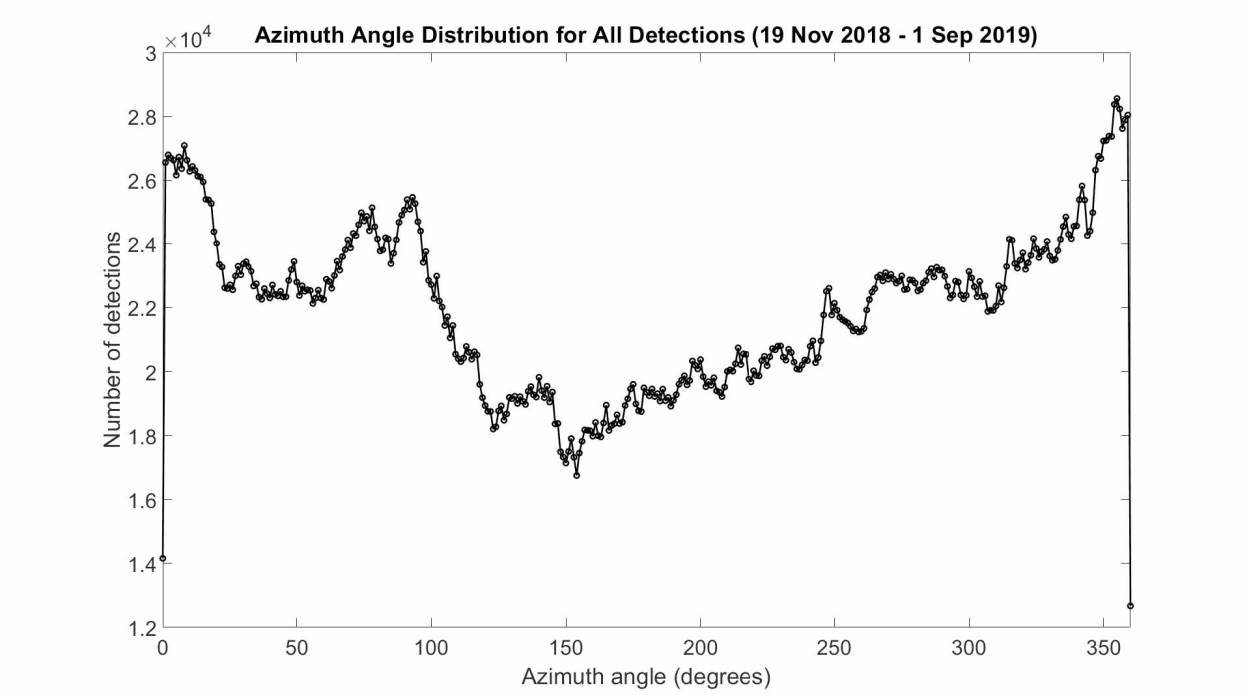

Figure 3.5: Azimuth angle distribution for all detections from Nov 19, 2018 through July 15, 2019. Detections are binned at 1 degree intervals. 0 degrees corresponds to east, 90 degrees to north, 180 degrees to west, and 270 degrees to south.

Figure 3.6 shows the distribution of azimuth angles for all detected objects, split up into 6 hour chunks throughout the day. It can be seen that the directions of peak detections changes throughout the day. As the Earth rotates, the radar moves in and out of the side of the Earth that is facing into space debris.

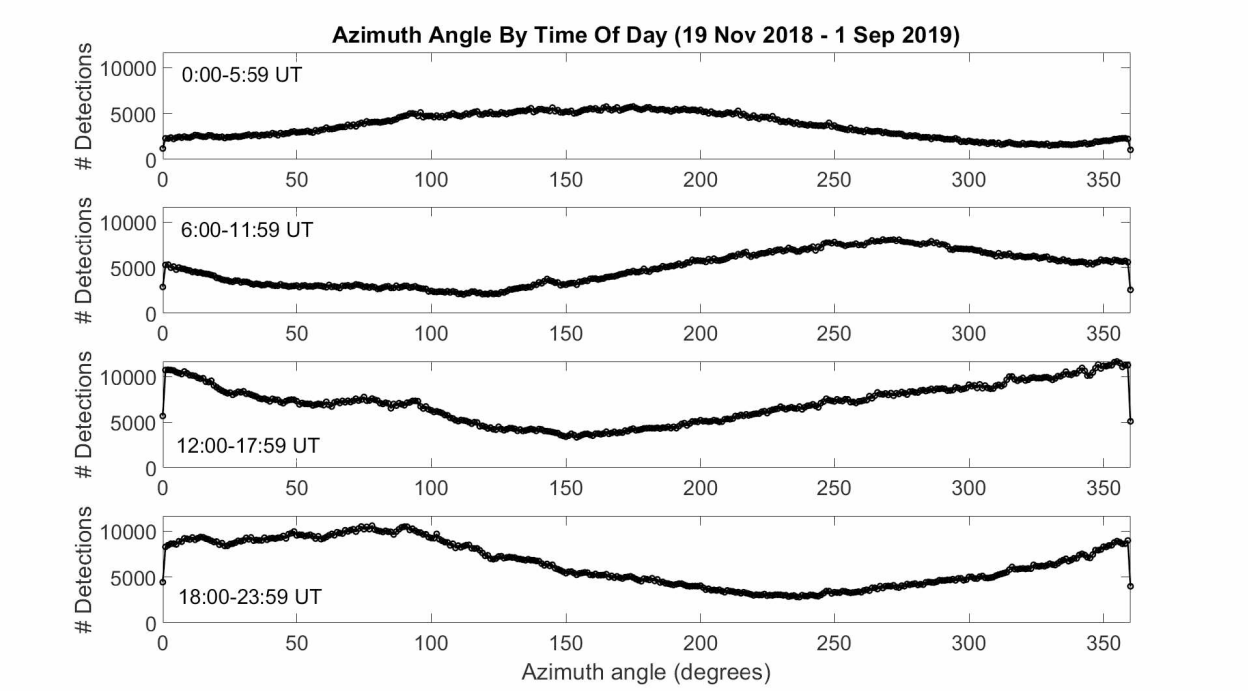

Figure 3.6: Azimuth angle split by time of day, grouped into 6 hour time blocks.

Figure 3.7 shows the distribution of signal-to-noise ratios (SNR) of all detected objects. The peak of the distribution is at 7 dB. The 4 dB SNR cutoff in the classification algorithm doesn't eliminate a majority of objects from the data product.

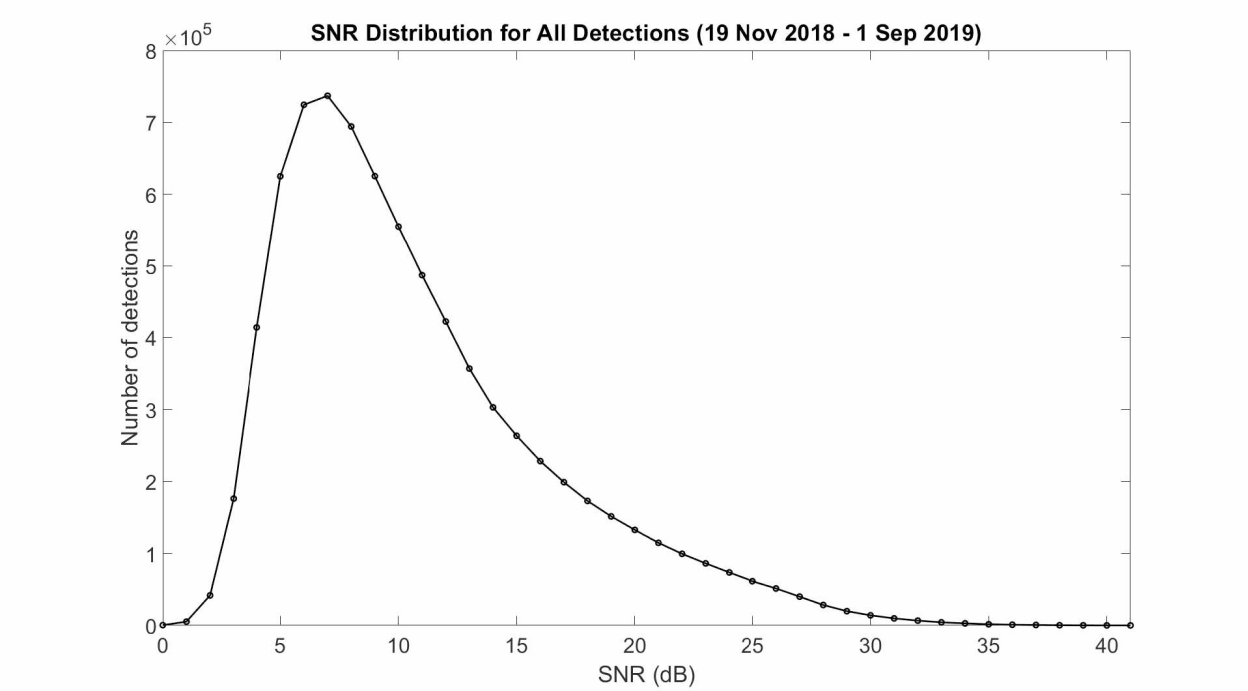

Figure 3.7: SNR distribution for all detections from Nov 19, 2018 through July 15, 2019. The detections are binned at 1 dB intervals. The peak is at 7 dB.

3.2 Hourly Wind Data

Middle atmospheric winds are the focus of many meteor radar studies. The method used by SKiYMET systems to calculate winds is described in *Hocking et al.* [2001]. The radial velocity of meteors is determined using the rate of change of the phase at zero lag of the sum of cross-correlation across antenna signals. A fit to lags -2, -1, 1, and 2 is used because of a spike in the cross-correlation function at zero lag due to correlated noise. After radial velocities have been calculated for all meteors, the processing software assumes a uniform wind $\mathbf{u} = (u, v, w)$ and then minimizes the quantity in Equation 3.1. In Equation 3.1, i refers to the meteor number in a certain time and height window, $\mathbf{r}_i^u$ is a vector pointing from the radar to the ith meteor trail, v_{ri} is the measured radial velocity, and $\mathbf{u} \cdot \mathbf{r}_i^u$ is a dot product of the two quantities.

$$\sum_i [\mathbf{u} \cdot \mathbf{r}_i^u - v_{ri}]^2 \tag{3.1}$$

Figures 3.8 and 3.9 show hourly zonal (east-west) and meridional (north-south) winds calculated over Poker Flat Research Range. Winds were calculated using 3 km altitude bins centered at altitudes of 82 km, 85 km, 88 km, 91 km, 94 km, and 98 km. A 168 hour (one week) smoothing window has been applied to the hourly wind data to eliminate tidal information and high frequency oscillations. There are several breaks in the wind data where the radar was down for maintenance.

The winds measured above Poker Flat Research Range have been compared qualitatively with winds measured above Esrange, Sweden (68°N, 21°E), using a meteor radar system [*Mitchell et al.*, 2002]. The zonal winds between the data sets at Poker Flat and Esrange agree fairly well. From December through May, eastward winds dominate from 82-98 km at speeds of about 10-20 m/s. From May to August, westward winds at speeds of about 20-30 m/s dominate from 82-90 km and eastward winds at speeds of about 20-30 m/s dominate from 90-98 km. In the winds measured above Poker Flat, there is a reversal of the zonal winds in mid December at altitudes of about 82-90 km. This is possibly associated with a Sudden Stratospheric Warming (SSW) event. The meridional winds also agree fairly well between the two data sets. The meridional component is mostly dominated by southward winds at speeds between 10-30 m/s for the entirety of the Poker Flat dataset (November through August).

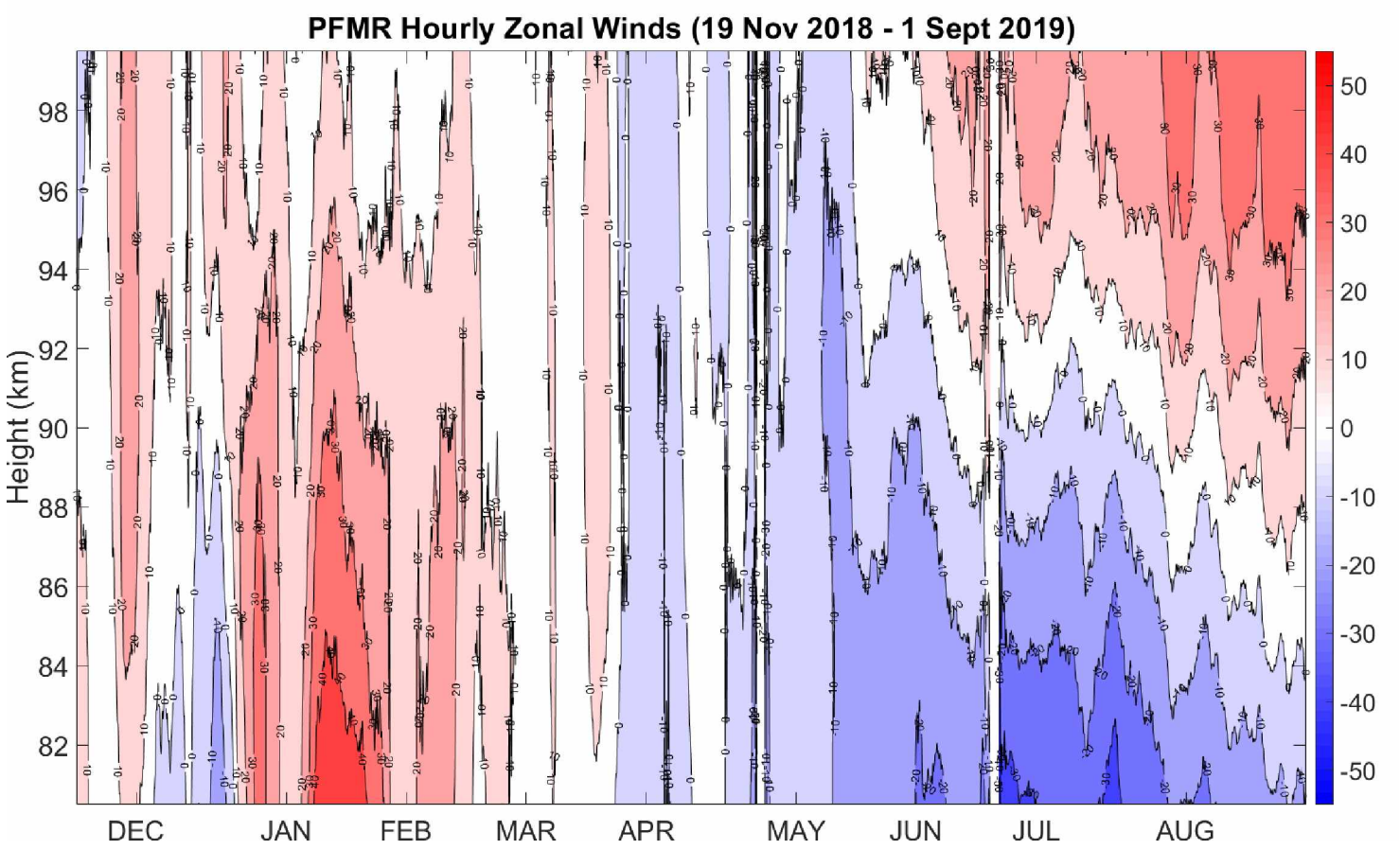

Figure 3.8: Hourly zonal wind data for Poker Flat Meteor Radar from Nov 19, 2018 through September 1, 2019. Red is eastward and blue is westward.

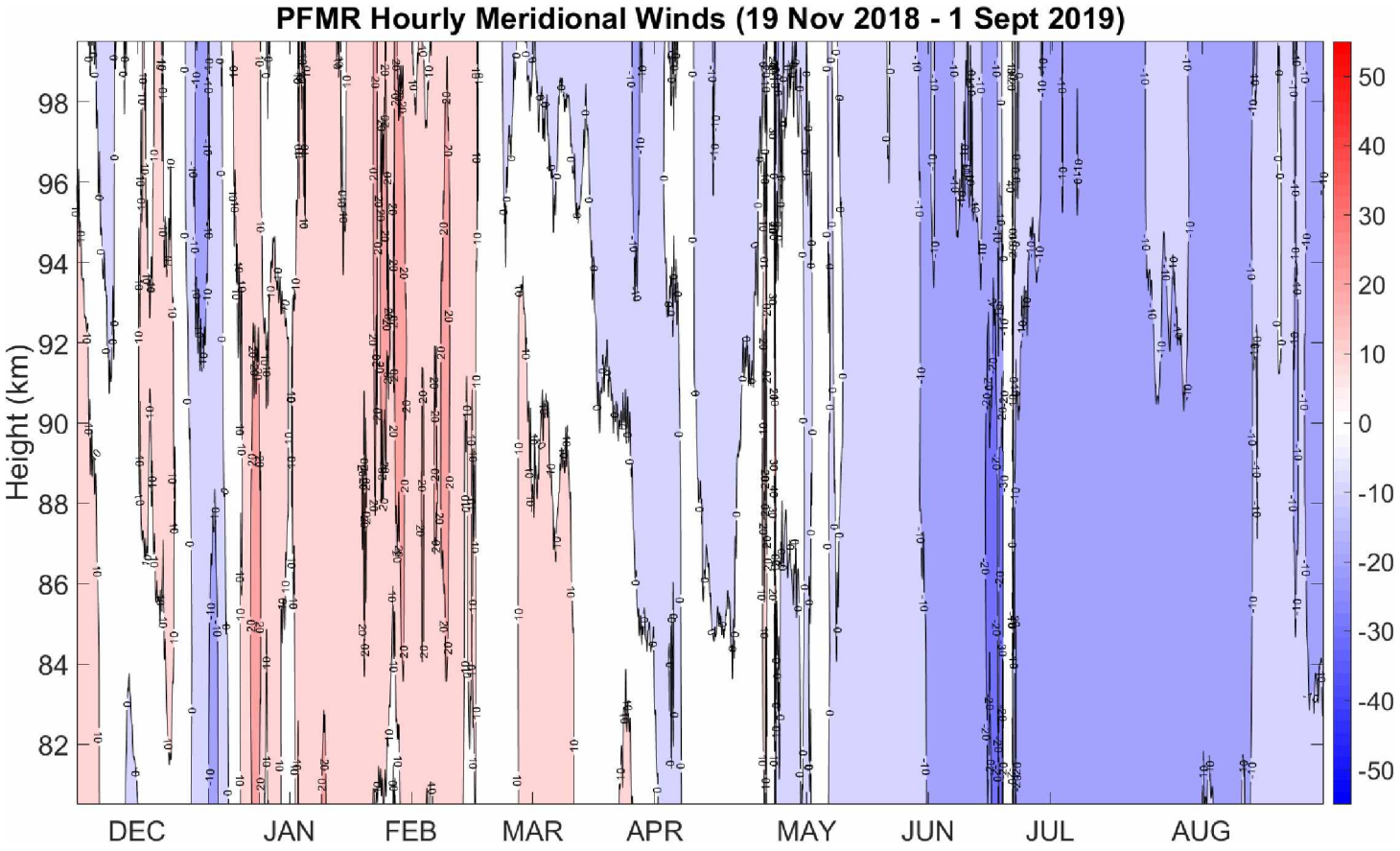

Figure 3.9: Hourly meridional wind data for Poker Flat Meteor Radar from Nov 19, 2018 through September 1, 2019. Red is eastward and blue is westward.

Chapter 4: Temperature Estimation Using Meteor Radar Observations: Methods and Current Issues

4.1 Estimating Temperature Using Meteor Decay Time

Meteor radar systems are routinely used for the measurement of winds in the mesosphere and lower thermosphere, under the assumption that the meteors are embedded in the background winds. In addition to wind measurements, meteor radar systems can be used to estimate middle atmosphere temperature. Early work in this area utilized a model relationship between temperature, ambipolar diffusion coefficient, and either density or pressure, as was done in *Greenhow and Hall* [1960]. Later methodologies developed used model densities or pressures to simplify the model relationship, as was done in *Hocking* [1999]. This chapter will present methodologies for estimating ambient mesospheric temperatures using meteor radar observations, along with some multi-instrument simultaneous observations aimed at testing these methodologies.

4.2 Estimating Temperature Without Density or Pressure Estimates

The power received from an underdense meteor, see Equation 4.1 [*McKinley*, 1961], is a function of the radar wavelength, λ, and the ambipolar diffusion coefficient, D. This allows for the ambipolar diffusion coefficient to be estimated using the observed decay time of an underdense meteor, as discussed in Chapter 1.

$$P_R(t) = P_0 \exp(\frac{-16\pi^2 D}{\lambda^2} t) \tag{4.1}$$

Jones [1995] showed that the ambipolar diffusion coefficient is related to the ambient temperature and pressure. Equation 4.2 shows this proportional relationship between the three quantities. This relationship comes from kinetic theory of ionized meteor trails and

assumes the meteor trail has a Gaussian ionization profile.

$$D \propto \frac{T^2}{P} \tag{4.2}$$

An alternative expression for the ambipolar diffusion coefficient in terms of temperature and pressure is given in Equation 4.3 [*Hocking et al.*, 1997], where q_e is the electron charge, k is the Boltzmann constant, P is the ambient pressure, and K_0 is a constant related to ion mobility in the plasma trail. If the diffusion coefficient and pressure are known, the temperature can be calculated.

$$D = \frac{2(1.013 \times 10^5) k K_0 T^2}{(273.16) q_e P} \tag{4.3}$$

A process for estimating temperature from only diffusion coefficient measurements is given in *Hocking* [1999]. This process starts by calculating a linear fit to the distribution of height versus log10(T_{un}) (T_{un} is defined in chapter 1) for all underdense meteor detections over the course of a day. The slope of these fits varies seasonally, indicating that it may be related to seasonal temperature variations. Using the relationship between the ambipolar diffusion coefficient, temperature, and pressure along with a model pressure expression for an isothermal atmosphere, shown in Equation 4.4, allows for the pressure term to be eliminated and the temperature to be estimated using only the ambipolar diffusion coefficient. In Equation 4.4, m is the mean mass of an atmospheric molecule, g is the acceleration due to gravity, k is the Boltzmann constant, and z is the altitude.

$$P = P_0 \exp(\frac{-mg}{kT} z) \tag{4.4}$$

According to *Hocking* [1999], an expression for temperature can be written to a reasonable approximation in terms of S_m, the slope of the linear fit to the height/inverse decay time

distribution, as shown below in Equation 4.5.

$$T \approx \frac{\log_{10} e\, mg}{k} S_m \approx 14.3 S_m \tag{4.5}$$

To improve accuracy, a temperature gradient can be incorporated into the expression rather than assuming an isothermal atmosphere. An expression for temperature as it changes with altitude is shown in Equation 4.6. In this expression, z' is defined as zero at the height of peak meteor activity, and α is defined in Equation 4.7, and is a function of the temperature gradient.

$$T = T_0(1 + \alpha z') \tag{4.6}$$

$$\alpha = \frac{1}{T_0}\frac{dT}{dz} \tag{4.7}$$

An alternative temperature gradient used in *Hocking* [1999] is shown in Equation 4.8, where θ is latitude, and # is the number of days from the summer solstice. This embodies a mean temperature gradient of -1.5K/km, which *Hocking* [1999] claims is reasonable. It is also common in the literature to use a temperature gradient calculated using a statistical atmospheric model such as MSIS.

$$\frac{dT}{dz} = -1.5 - [-2.5\, exp(-(\theta - 45)^2/200) + 1.5\, exp(-(\theta - 90)^2/1350)]\, exp(-\#^2/3200) \tag{4.8}$$

A model expression for pressure in a non-isothermal atmosphere is shown in Equation 4.9.

$$P = P_0\, exp(-\int_0^{z'} \frac{mg}{kT(z)} dz'') \tag{4.9}$$

Between the ambipolar diffusion coefficient slope and the temperature is now a function of the temperature gradient (see Equation 4.10).

$$T_0 = S_m(2\frac{dT}{dz} + \frac{mg}{k}) \log_{10} e \tag{4.10}$$

There are several limitations to this method. The resulting temperature estimate is for the height of peak meteor activity and is usually utilized to estimate average temperature over one day. Another issue is that different slopes are calculated depending on how the linear fit is done (height vs log10(T_{un}) or log10(T_{un}) vs height). *Kim et al.* [2012] found that using height as the independent variable is more appropriate. There is also an issue with the number of meteors at the peak of the height distribution skewing the fit. This can be mostly avoided by binning the height/T_{un} distribution and using the median height and decay time in each bin for the fit. This method for temperature estimation is convenient because it allows for temperature estimates using only measurements from the meteor radar. However, if estimates for pressure or density are available from other instruments, it may be possible to estimate temperature more accurately and with better temporal and spatial resolution.

4.2.1 Estimating Temperature With Density or Pressure Estimates

Similarly to the relationship between ambipolar diffusion coefficient, temperature, and pressure, there also exists a related relationship between ambipolar diffusion coefficient, temperature, and density. This is shown in Equation 4.11 and is found in *Kaiser* [1953].

$$D \propto \frac{T^{1/2}}{\rho} \tag{4.11}$$

The ambipolar diffusion coefficient can be written as a function of ambient density and temperature *Greenhow and Neufeld* [1955], where μ is the mean mass of a diffusing particle, σ_i is the collision diameter for ionized meteor atoms and air molecules, and n is the ambient particle density. This is shown below in Equation 4.12.

$$D = \frac{3.5}{\pi n \sigma_i}\left(\frac{kT}{\pi\mu}\right)^{1/2} \tag{4.12}$$

Additionally, the ambipolar diffusion coefficient can be expressed in terms of ambient mass density, ρ, instead of particle density [*Greenhow and Hall*, 1960], as shown in Equation

4.13. A is the collision cross section, and m is the mean mass of an air molecule. If both the ambipolar diffusion coefficient and ambient density are known, the temperature can be calculated.

$$D = \frac{3.5m}{4\rho A}\left(\frac{kT}{\pi\mu}\right)^{1/2} \tag{4.13}$$

A succinct expression of these relationships can be found in *Younger et al.* [2015], where the most common values for most of the constants are filled in to simplify things. This is shown in Equation 4.14, where K_0 is a constant related to the mobility of ions in the meteor trail and has units of m^2/sV. Note that the temperature is not raised to the 1/2 in this expression. *Jones* [1995] notes that the exponent on the temperature term may be either 1 or 1/2. That ambiguity still exists in the literature today, so both cases are examined in this chapter.

$$D = 6.39 \times 10^{-2} K_0 \frac{T^2}{P} = 2.23 \times 10^{-4} K_0 \frac{T}{\rho} \tag{4.14}$$

An assumption throughout all of these methods for temperature estimation is that ambipolar diffusion is the primary driver of meteor trail dissipation. For short-lived trails, this is likely true, but other effects can have significant impacts on de-ionization of longer lasting meteors. For this reason only underdense meteors are typically used for temperature estimates from meteor radar systems.

4.3 Observations Over Sodankylä Geophysical Observatory

A meteor radar operating at Sodankylä Geophysical Observatory (SGO) in northern Finland (67.4°,26.6°) was used to estimate average daily temperatures from 01 October 2017 through 30 December 2017 (shown in Figure 4.1). The temperatures were calculated using the method given in *Hocking* [1999] and using a temperature gradient calculated using MSIS [*Kozlovsky et al.*, 2016]. The daily average temperatures in Figure 4.1 were provided along with one day of raw radar data (13 December 2017) courtesy of Alexander Kozlovsky, SGO. A large temperature dip can be seen in December, corresponding with the Geminids

meteor shower. The same temperature dip can be seen using data from SGO nearly every year from 2008-2014 [*Kozlovsky et al.*, 2016]. In addition to calculating daily average radar temperatures, *Kozlovsky et al.* [2016] compared these temperatures with measurements from the Aura satellite. They concluded that the temperature dip seen during the Geminids was not a real temperature anomaly and was likely due to an issue with data processing.

Using the day of raw radar data that was provided, the temperature was calculated using all detections and then again using only detections that were determined to be underdense meteors using the method discussed in Chapter 2 from *Holdsworth et al.* [2004] and compared to the daily temperatures that were provided. The calculated temperature using all detections on 13 December 2017 was 163.66 K which matches the temperature provided for that day. All of the echoes recorded for that day were categorized and of the 12,646 echoes only 5712 were found to be underdense. The temperature was re-calculated using only the underdense meteors and was 199.56 K. This higher temperature seems to be more in line with the rest of the time series, and suggests that using only underdense meteors for temperature estimation may remove the temperature dip seen during the Geminids. *Kozlovsky et al.* [2016] hypothesizes that there is a larger amount of overdense meteors during the meteor shower which skews the daily temperature estimation.

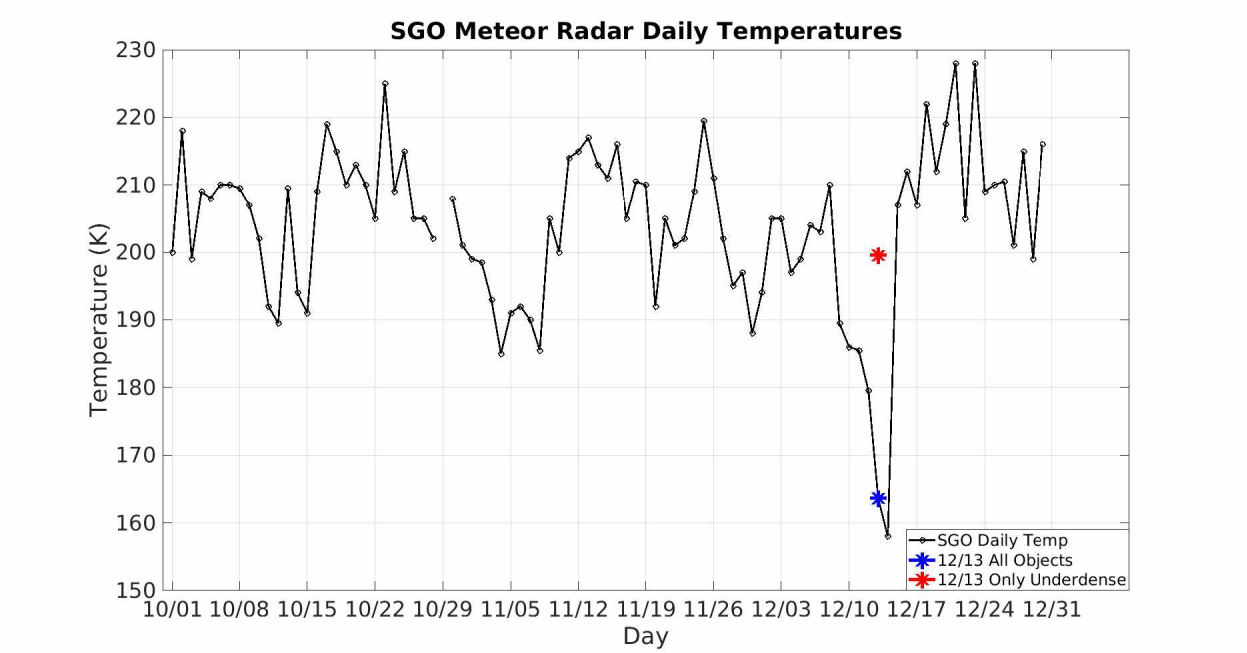

Figure 4.1: Daily average temperatures observed with meteor radar over Sodankylä Geophysical Observatory. The temperature calculated with only underdense meteors (red) is much closer to the rest of the time series than the temperature calculated using all detections for the day (blue).

4.4 Simultaneous Observations with Multiple Instruments

Simultaneous observations were carried out at Poker Flat on December 23-24, 2018 between a meteor radar, a Rayliegh lidar, and a Sodium Resonance Wind Temperature lidar (SRWTL). The meteor radar estimates the ambipolar diffusion coefficient, the Rayliegh lidar estimates neutral density, and the SRWTL estimates neutral temperature. All measurements presented from these instruments are from 80-100 km. Ideally, using two of these measurements would allow for the third parameter to be estimated. Using these three instruments allows for the self-consistency of the ambipolar diffusion-temperature-density model to be checked. The next sections examine multi-instrument observations on December 23 and 24, 2018.

4.4.1 Observations on 23 December 2018

The meteor radar, Rayleigh lidar, and SRWTL simultaneously collected data on December 23, 2018 from 2:20 to 15:55 UT. Figure 4.2 shows an average temperature profile as estimated by the SRWTL, along with a temperature estimate from the meteor radar. The temperature was estimated from meteor radar data using the method discussed in section 4.1.1 of this thesis using a temperature gradient of -2.48 K/km calculated from MSIS. Only underdense meteors (as determined by the classification algorithm discussed in Chapter 2) were used, and only those between 80 to 100 km and between 2:20 and 15:55 UT. This average temperature estimate from the radar was placed at 91 km, the height of peak meteor activity for that day. There is slight disagreement between the SRWTL temperature estimate at 91 km, 199.95 K, and the meteor radar average temperature, 211.59 K, a 6% over-estimation. However, note that the uncertainty on the radar estimate is ±7.45 K and the uncertainty on the SRWTL estimate is ±9.89 K, so there is some overlap.

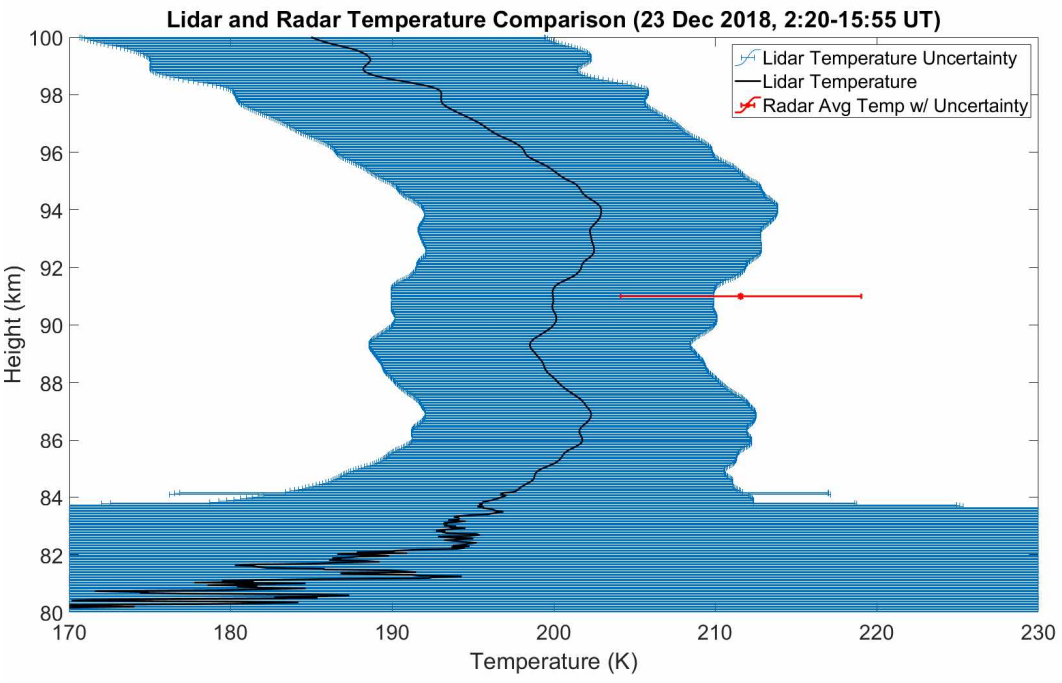

Figure 4.2: Average temperature profile from the SRWTL (black) and their uncertainty (blue) are shown alongside the radar estimate for temperature using the method discussed in section 4.1.1, along with its uncertainty (red).

Given simultaneous measurements between three instruments, it is possible to test the hypothesis that Equation 4.14 can be used to calculate temperature given the ambipolar diffusion coefficient and density. Figure 4.3 shows an ambipolar diffusion coefficient profile estimated by the meteor radar, as well as a neutral density profile estimated by the Rayleigh lidar for 2:20 to 15:55 UT on December 23. The ambipolar diffusion coefficient was calculated using the decay times of the meteors and is an average over the time the lidars were operating, binned at 1 km. It generally follows an expected trend by increasing sort of exponentially as altitude increases [*McKinley*, 1961]. The neutral density profile is also an average over the running time of the lidars and binned at 1 km. It increases exponentially as altitude decreases, as expected.

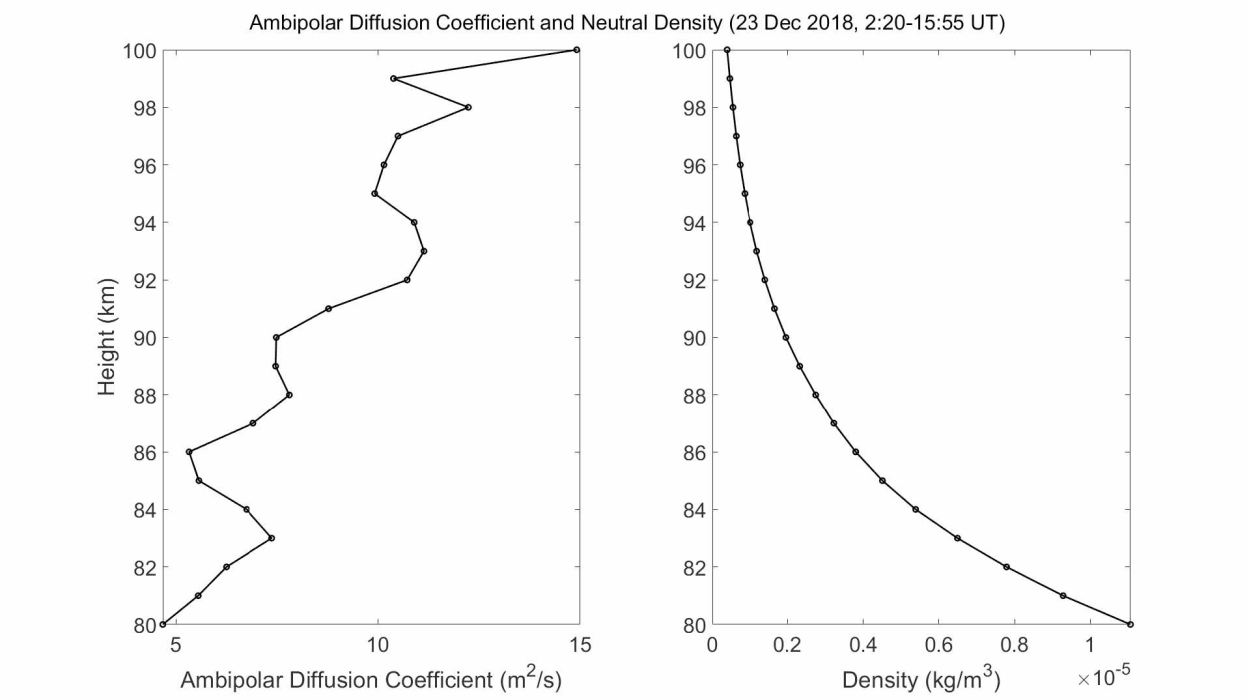

Figure 4.3: The ambipolar diffusion coefficient estimated by the meteor radar is shown on the left side of the figure. The neutral density estimated by the Rayleigh lidar is shown on the right side of the figure.

It is important to note that while the ambipolar diffusion coefficient average profile was used in the following calculations, there is quite a bit of variation in ambipolar diffusion

coefficients in each height bin. Figure 4.4 shows a scatter plot of the ambipolar diffusion coefficient with height for the 4142 underdense meteors used in these calculations.

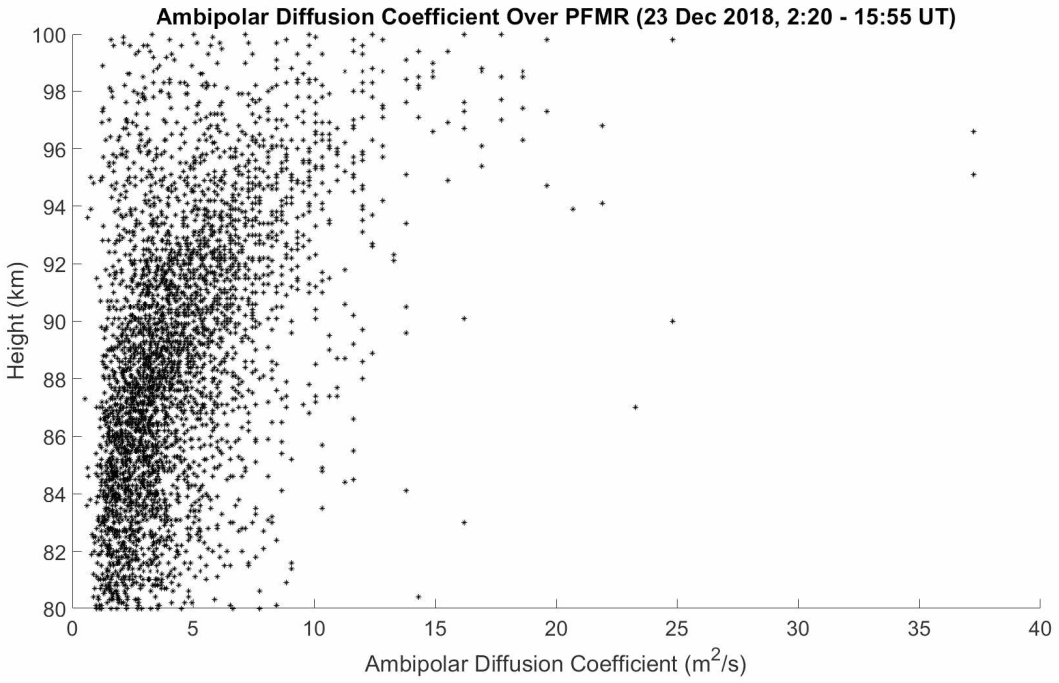

Figure 4.4: Ambipolar diffusion coefficient as calculated for each of the 4142 underdense meteors detected during the observation window is plotted against height of the detection. There is significant variability in each height bin.

The ambipolar diffusion coefficient estimates from the meteor radar and the neutral density estimates from the Rayleigh lidar were used to calculate a temperature profile using the method shown in section 4.1.2 of this thesis. Equation 4.14 was used to calculate the temperature profile, using a K_0 value of $2.5\times10^{-4}\frac{m^2}{sV}$, as suggested in *Younger et al.* [2015]. Figure 4.5 shows the result of this calculation compared to the average radar temperature calculated using the method discusses earlier in this section, along with the SRWTL temperature profile. The radar average temperature of 211.59 K is 23% lower than the new temperature estimate at 91 km of 261.08 K. It is clear that using Equation 4.14 with the values provided does not result in accurate temperature calculations. It may be that there is something missing in the model that needs to be accounted for, such as chemistry or mixing due to turbulence.

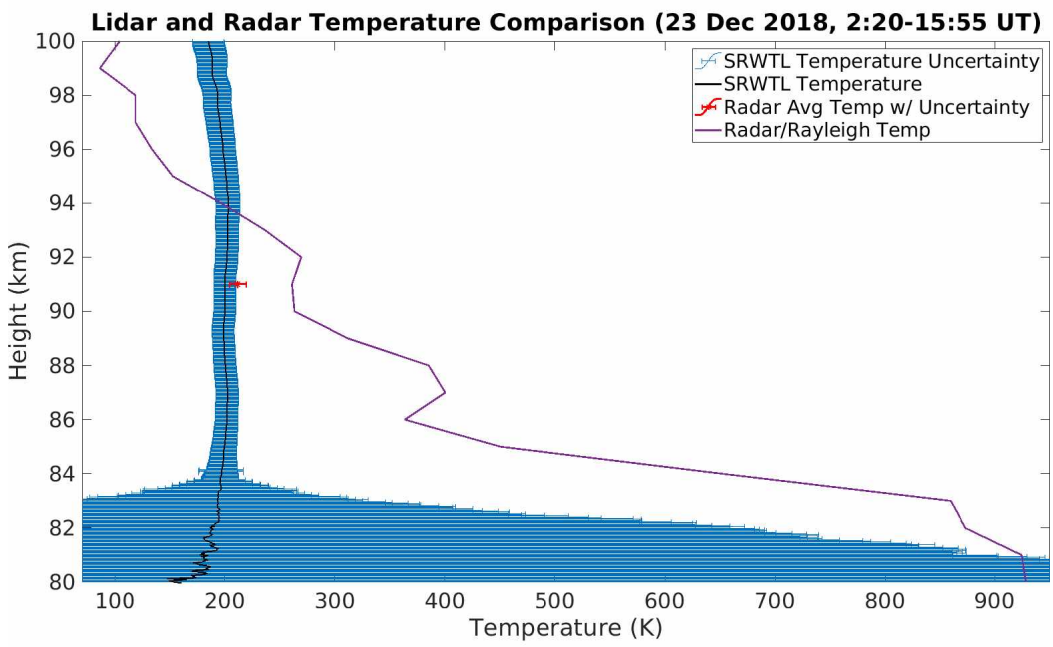

Figure 4.5: The temperature estimate from the radar using the method shown in section 4.1.1 (red) is compared to the temperature estimate using the ambipolar diffusion coefficient and the neutral density, as shown in section 4.1.2 (purple). The SRWTL temperature profile is shown in black with blue error bars.

A temperature profile was also calculated using 4.13. The ambipolar diffusion and density profiles were taken from the radar and Rayleigh lidar systems. The values for mean mass of an air molecule, m, and mean mass of a diffusing particle, μ, were taken from *Greenhow and Hall* [1960] as 26 amu and 35 amu, respectively. The collision cross section, A, was calculated using an expression from *Strelnikova et al.* [2007] and collision frequencies from *Hill and Bowhill* [1977]. Equations 4.15 and 4.16 are for collision cross section, A, and are from *Strelnikova et al.* [2007]. In these equations, m_p is the mass of a diffusing particle in kilograms, m_n is the mass of an atmospheric particle in kilograms, k is the Boltzmann constant, T is the temperature in K, r_p is the radius of a diffusing particle in meters, r_n is the radius of an atmospheric particle in meters, and ν_{pn} is the total collision frequency in s^{-1}. Equation 4.17 is for collision frequency, ν_{pn}, and is from *Hill and Bowhill* [1977]. In this equation, μ is the mass of a diffusing particle in amu, and N_n is the neutral number density

in cm^{-3}. A was calculated at 1 km height intervals using neutral number densities from the Rayleigh lidar using these equations.

$$A = \pi(r_p + r_n)^2 \tag{4.15}$$

$$(r_p + r_n)^2 = \frac{(3\nu_{pn}(m_p + m_n))}{8m_p N_n}\sqrt{\frac{m_p}{2\pi k T m_n(m_p + m_n)}} \tag{4.16}$$

$$\begin{aligned}\nu_{pn} = 2.6 \times 10^{-9} N_n \Bigg(0.78\frac{28}{\mu + 28}\sqrt{\frac{1.74(\mu + 28)}{28\mu}} + 0.21\frac{32}{\mu + 32}\sqrt{\frac{1.57(\mu + 32)}{32\mu}} \\ + 0.01\frac{40}{\mu + 40}\sqrt{\frac{1.64(\mu + 40)}{40\mu}}\Bigg)\end{aligned} \tag{4.17}$$

The temperature profile calculated using Equation 4.13 was compared to the temperature profile calculated using Equation 4.14, and the radar average temperature estimate. The results are shown in Figure 4.6. The radar average temperature is 211.6 K (red), the temperature at 91 km from Equation 4.14 is 261.1 K (purple), and the temperature at 91 km from Equation 4.13 is much higher at 3172.4 K.

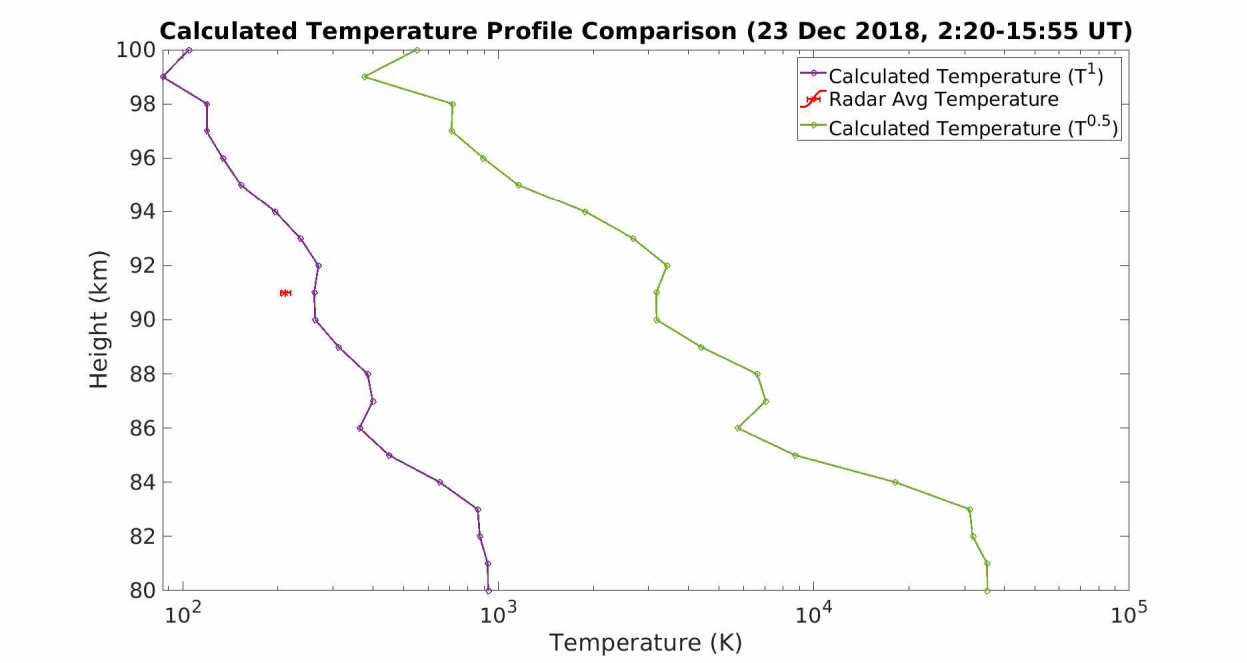

Figure 4.6: A temperature profile was calculated for 23 December 2018 using Equation 4.13 (green) and compared to the other temperatures calculated using the radar data (red and purple).

4.4.2 Observations on 24 December 2018

Multi-instrument, simultaneous observations were again carried out on December 24, 2018 from 4:30 to 18:15 UT. Figure 4.7 shows an average temperature profile estimated by the SRWTL, and the average temperature estimated from the meteor radar data. The same analysis was carried out for this night as for the previous night. Only underdense meteors were used between 4:30 and 18:55 UT. The height of peak meteor activity was again at 91 km. The SRWTL temperature estimate at 91 km, 207.04 K, and the average radar temperature, 206.74 K, only disagree by 0.58%.

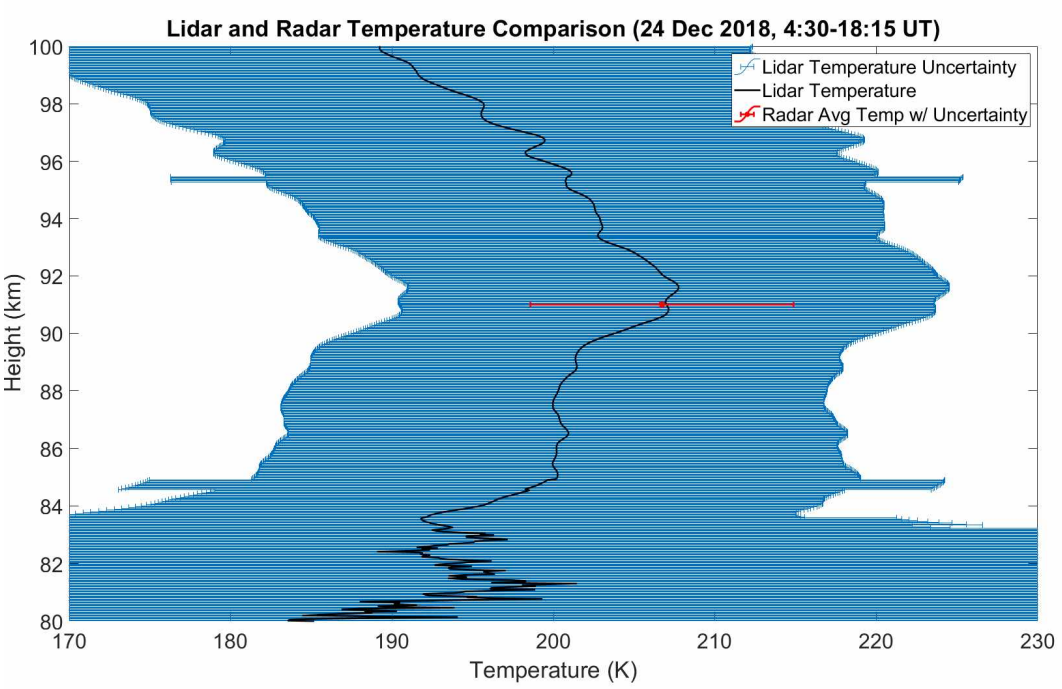

Figure 4.7: Average temperature profile from the SRWTL (black) and their uncertainty (blue) are shown alongside the radar estimate for temperature using the method discussed in section 4.1.1 along with its uncertainty (red).

Similarly to the observations on December 23, measurements from all three instruments are used to test the hypothesis that temperature can be calculated using the ambipolar diffusion coefficient and density. Figure 4.8 shows an ambipolar diffusion coefficient profile estimated by the meteor radar and a neutral density profile estimated by the Rayleigh lidar for 4:30 to 18:55 UT on December 24, 2018. Both profiles are averages over the running time of the lidar and binned at 1 km. Again, both profiles generally follow expected trends with height.

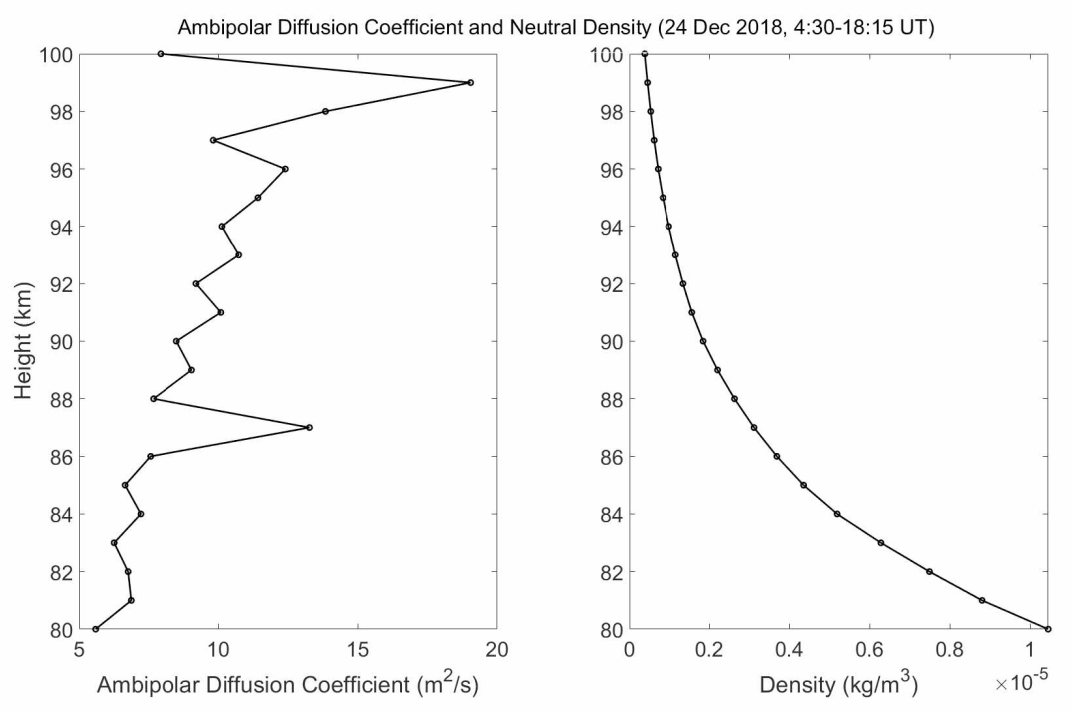

Figure 4.8: The average ambipolar diffusion coefficient estimated by the meteor radar is shown on the left side of the figure. The average neutral density estimated by the Rayleigh lidar is shown on the right side of the figure.

Similar to the previous day of multi-instrument observations, there is a lot of variation the in ambipolar diffusion coefficient in each height bin. Figure 4.9 shows a scatter plot of the ambipolar diffusion coefficient against height of detection for the 4409 underdense meteors used in these calculations.

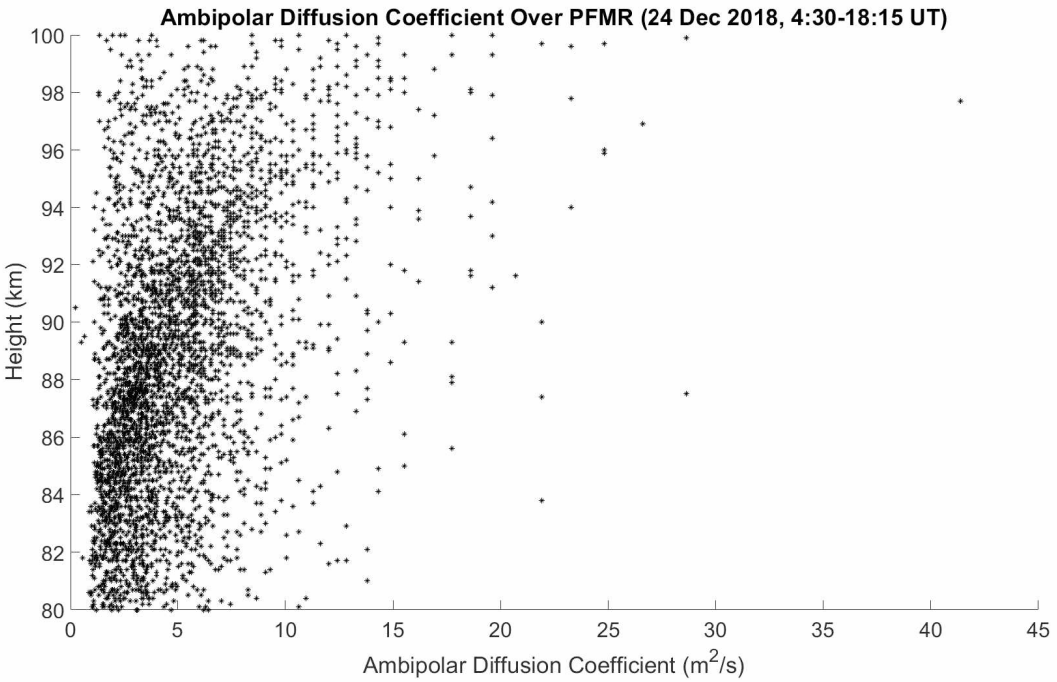

Figure 4.9: Ambipolar diffusion coefficient as calculated for each of the 4409 underdense meteors detected during the observation window is plotted against height of the detection. There is significant variability in each height bin.

Similar to the analysis of the previous day's data, the ambipolar diffusion coefficient profile and the neutral density profile were used to calculate a temperature profile using Equation 4.14. The calculated temperature at 91 km is 283.58 K, and the radar average temperature estimate is 207.04 K. Again, there is significant disagreement between these two temperature estimation methods.

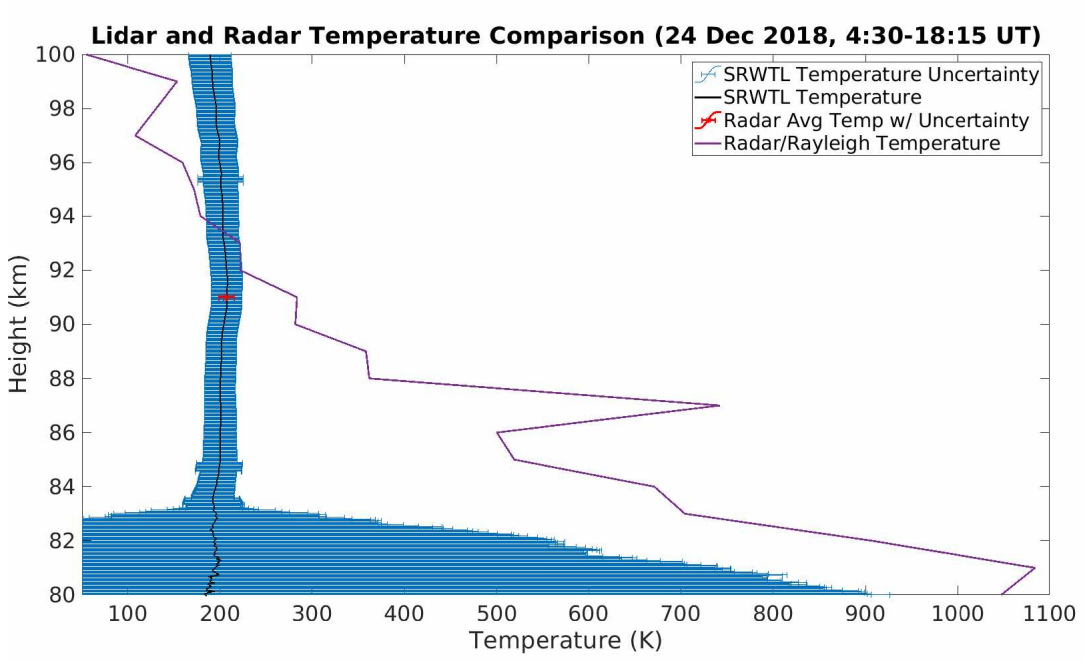

Figure 4.10: The temperature estimate from the radar using the method shown in section 4.1.1 (red) is compared to the temperature estimate using the ambipolar diffusion coefficient and the neutral density, as shown in section 4.1.2 (purple). The SRWTL temperature profile is shown in black with blue error bars.

A second temperature profile was calculated using Equation 4.13 and compared to the other calculated temperature profile from Equation 4.14. The results are shown in Figure 4.11. The radar average temperature is 206.7 K (red), the temperature at 91 km from Equation 4.14 is 283.6 K (purple), and the temperature at 91 km from Equation 4.13 is once again much higher at 3742.8 K.

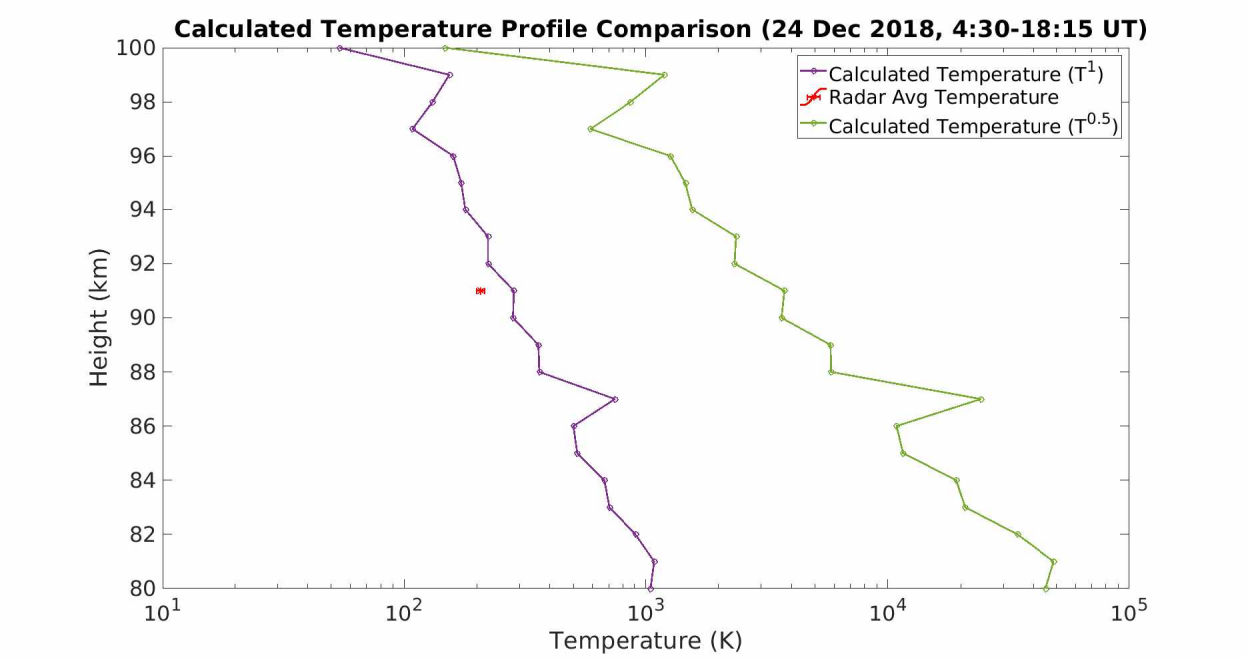

Figure 4.11: A temperature profile was calculated for 24 December 2018 using Equation 4.13 (green) and compared to the other temperatures calculated using the radar data (red and purple).

On both days of simultaneous observations, it is clear that there does not exist a scalar value that will result in the temperature profile from the SRWTL and the temperature profile from the radar and Rayleigh lidar to be approximately equal to each other. If Equation 4.13 is rewritten so that all of the constant terms are collapsed into a single term, as in 4.18, the value of the constant "B" that results in a calculated temperature profile that is exactly equal to the SRWTL temperature profile can be calculated.

$$D = B\frac{T^{1/2}}{\rho} \tag{4.18}$$

$$D = B\frac{T}{\rho} \tag{4.19}$$

Figure 4.12 shows the calculated value of "B" that makes the calculated temperature using data from multiple instruments work out to be exactly the same as the SWRTL temperature profile. For both days of simultaneous observations, similar values of "B" are needed to

make the calculations work out. There is clearly something that is not being accounted for in either the model equation between temperature, density, and ambipolar diffusion, or in one or more of the instrument measurements.

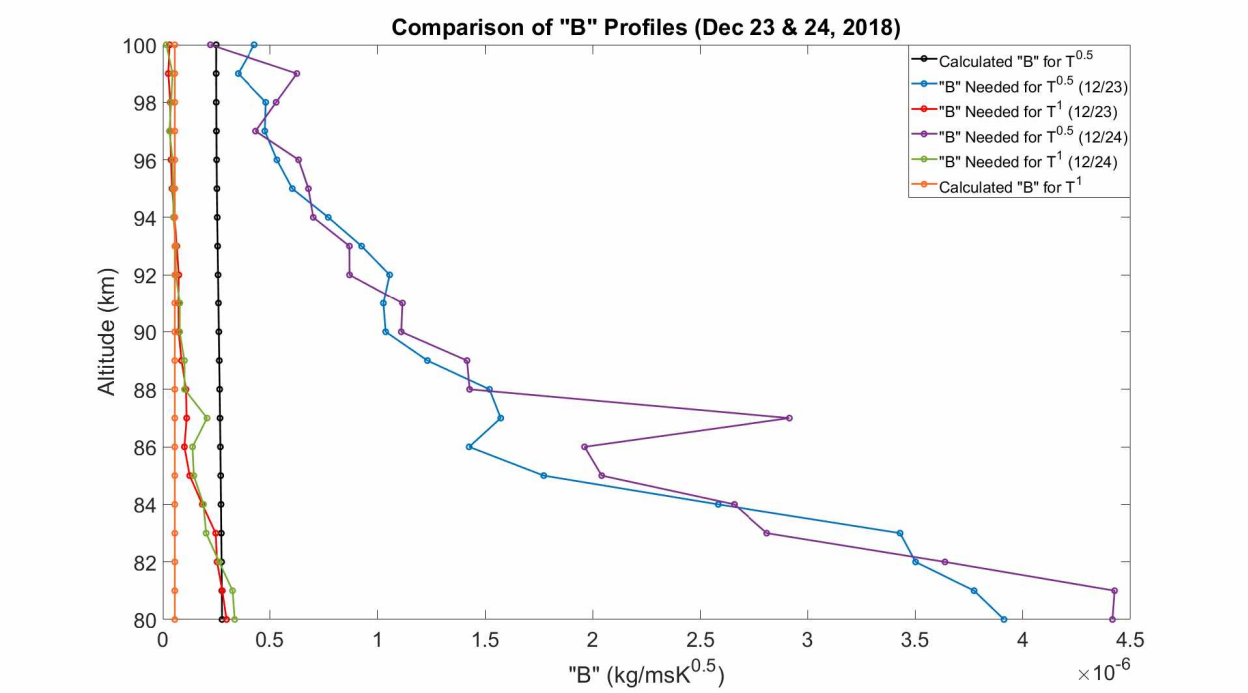

Figure 4.12: The value of "B" from Equation 4.18 needed to force the calculated temperature match the SRWTL temperature profile exactly. The value of "B" needed on December 23 and 24 for Equation 4.13 is shown in blue and purple, respectively. The value of "B" needed for December 23 and 24 for Equation 4.14 is shown in red and green, respectively. The calculated value of "B" used in calculations in this chapter is shown in black for the $T^{0.5}$ model and orange for the T^1 model.

It may be that there are chemical effects that are significant drivers of meteor trail decay under certain conditions as stated in *Jones et al.* [1990] and shown in Figure 1.3. Figure 4.13 shows the decay times for all meteor detections from 19 November 2018 through 01 September 2019 above Poker Flat Meteor Radar, as well as the decay times for only the detections assumed to be underdense meteors. The same complementary cumulative distribution discussed in Chapter 1 is used to illustrate the inflection point that is not expected if meteor trails are dissipated by ambipolar diffusion alone. As discussed in Chapter 1, the inflection point in the distribution of meteor decay times for meteor trails affected by

only ambipolar diffusion is not expected [*Ceplecha et al.*, 1998]. Clearly there are drivers of meteor trail decay that affect longer duration meteor trails that are not well understood and not accounted for in the ambipolar diffusion decay model.

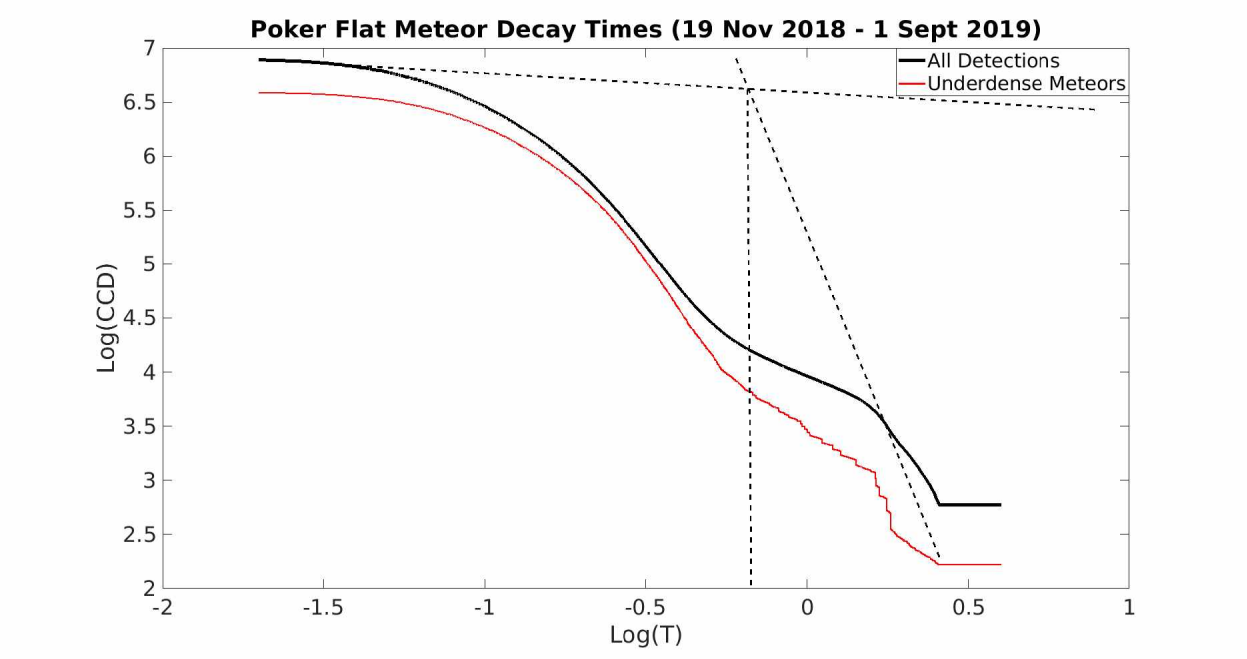

Figure 4.13: CCD of decay times of all meteor detections from 19 Nov 2018 through 01 Sept 2019 above PFMR (black). The same curve using only the detections assumed to be underdense meteors is shown also (red). There is an inflection point in the curve, as expected from *Jones et al.* [1990].

Cevolani et al. [1999] uses the inflection point in this type of plot to estimate ozone concentration in the altitude region of 80-100 km (see Equation 4.20.) In this equation, T_c is the "critical value" of echo duration, or where the inflection point in the distribution is, and α is the time constant for oxidizing meteor ions by ozone.

$$[O_3] = (\alpha T_c)^{-1} \tag{4.20}$$

An average ozone concentration was calculated using Figure 4.13 and Equation 4.20. A value of $2\text{x}10^{-16}$ m^3s^{-1} was used [*Baggaley*, 1972], along with the T_c value of 664.0 ms taken from the figure. The result was an ozone concentration $[O_3]$ of $7.5\text{x}10^{15}$ m^{-3}. *Hocking et al.*

[2016] and *Cevolani et al.* [1999] both report annual average ozone concentration in the 80-100 km region to be on the order of 10^{13}-10^{14} m^{-3}, so this estimate may be a bit high. It only uses data from November through September, and it is using all detections, not just underdense and overdense meteors.

4.4.3 Summary of Simultaneous Observations

In this chapter, neutral temperatures were calculated using meteor radar data, along with data from a SRWTL, and a Rayleigh lidar. The radar average temperatures seemed to agree with the SRWTL temperatures at 91 km very well. These daily average temperatures are the most commonly reported meteor radar temperature estimates in the literature. The hypothesis that temperature can be calculated using the ambipolar diffusion coefficient and density along with a model equation clearly needs further testing. It is also likely that the model relationship needs adjusting.

Chapter 5: Conclusions and Future Work

5.1 Introduction

This chapter will summarize work done for this thesis and outline some of the current issues in meteor radar classification and signal processing. The work conducted to complete this thesis has highlighted the need for future work in several areas, including classification of overdense meteors in radio backscatter, signal processing techniques for long-duration meteors, and temperature estimation using meteor radar data.

5.2 Meteor Radar Interferometry

Two methods of meteor radar interferometry were compared in this thesis using a set of 2500 simulated meteors. The Jones method seems to be incredibly robust, but it does not provide a measure of uncertainty for the calculated direction cosines. The Vaudrin model fit method allows for statistical uncertainties for fitted parameters, but does not seem to be very stable. This may be due to the number of parameters that are being fit to, or it may be issues inherent in the Levenberg-Marquardt optimization algorithm when applied to high-dimensional data. Future work in this area could include developing more stable and robust methods of estimating statistical uncertainties for direction cosines of meteor detections.

5.3 Classification of Meteors

Three methods of meteor backscatter classification were compared in this thesis using a set of 307 visually classified meteor signals. The time series classification based on *Holdsworth et al.* [2004] seems to be the most robust, classifying about 91% of underdense meteors correctly and 89% of other detections correctly. The major issue with this classification scheme is the 42% of overdense meteors that were incorrectly classified as underdense meteors. There is not currently a signal model for overdense meteor backscatter in the same sense that there is for underdense meteors. This makes it difficult to build a classification algo-

rithm for overdense meteors that is similar to algorithms for underdense meteors like the one in *Holdsworth et al.* [2004]. Machine learning classification techniques may be a way to accurately classify overdense meteor echoes, but as shown in Chapter 2, significantly more time needs to be spent tuning the machine learning model to recognize overdense meteors. The large variability in overdense meteor backscatter makes this difficult. The duration and oscillatory behavior of overdense meteor scatter can vary greatly between meteors, posing some challenges to machine learning techniques that look for specific signal parameters.

5.4 Signal Processing for Long-Duration Meteors

Long-duration meteors tend to be overdense, but long lasting underdense echoes are also possible. As the duration of the echo increases, there are additional effects beyond ambipolar diffusion that start to drive the decay of the trail. In these cases, the traditional meteor decay model that is driven only by ambipolar diffusion is no longer adequate, and does not account for all of the physics that is required for accurate parameter estimation from these echoes. Future work in this area could include developing new models for meteor decay that include effects other than ambipolar diffusion. The work done in this thesis on both classification and temperature estimation shows that there is likely physics that is not being accounted for in these models.

5.5 Meteor Radar Temperature Estimates

This thesis used data from Sodankylä Geophysical Observatory to show the effect of meteor classification on temperature estimation. In addition, multi-instrument data from Poker Flat Research Range was used to investigate the self-consistency of two model relationships between the ambipolar diffusion coefficient, density, and temperature. One model seems to produce results that are better than the other, but neither seems to work very well. There is still a lot of work to be done in the area of meteor radar temperature estimates. While the average temperature estimates using the method shown in section 4.1.1 of this thesis agree

reasonably well with lidar temperature estimates at altitudes near 90 km, it would be nice to have a temperature profile across the entire meteor region. As stated previously, it is possible that the models are not accounting for some of the physics that is driving meteor trail decay. It is also possible that meteor selection criteria for these temperature estimate analyses needs to be revised, as some of the meteors may not adhere to the ambipolar diffusion model that most of the processing is based on.

References

Baggaley, W. (1978), The de-ionizatton of dense meteor trains, *Planetary and Space Science*, *26*(10), 979–981, doi:10.1016/0032-0633(78)90080-6.

Baggaley, W. (1979), The interpretation of overdense radio meteor echo duration characteristics, *Astronomical Institutes of Czechoslovakia, Bulletin*, *30*(3), 184–189.

Baggaley, W. J. (1972), The Effect of Meteoric Ion Processes on Radio Studies of Meteoroids, *Monthly Notices of the Royal Astronomical Society*, *159*(2), 203–217, doi:10.1093/mnras/159.2.203.

Ceplecha, Z., J. Borovička, W. G. Elford, D. O. ReVelle, R. L. Hawkes, V. Porubčan, and M. Šimek (1998), Meteor Phenomena and Bodies, *Space Science Reviews*, *84*(3/4), 327–471, doi:10.1023/A:1005069928850.

Cevolani, G., and G. Pupillo (2003), Ground-based radio observations to probe the ozone content in the meteor region, *Annals of Geophysics*, (46).

Cevolani, G., A. Hajduk, M. Hajduková, V. Porubčan, and G. Trivellone (1999), Ozone concentration at meteor heights determined from forward-scatter radar echoes, *Journal of Atmospheric and Solar-Terrestrial Physics*, *61*(7), 539–543, doi:10.1016/S1364-6826(99)00019-X.

Greenhow, J., and J. Hall (1960), Diurnal variations of density and scale height in the upper atmosphere, *Journal of Atmospheric and Terrestrial Physics*, *18*(2-3), 203–214, doi:10.1016/0021-9169(60)90093-3.

Greenhow, J., and E. Neufeld (1955), The diffusion of ionized meteor trails in the upper atmosphere, *Journal of Atmospheric and Terrestrial Physics*, *6*(1-6), 133–140, doi:10.1016/0021-9169(55)90020-9.

Hill, R., and S. Bowhill (1977), Collision frequencies for use in the continuum momentum equations applied to the lower ionosphere, *Journal of Atmospheric and Terrestrial Physics*, *39*(7), 803–811, doi:10.1016/0021-9169(77)90143-X.

Hocking, W., B. Fuller, and B. Vandepeer (2001), Real-time determination of meteor-related parameters utilizing modern digital technology, *Journal of Atmospheric and Solar-Terrestrial Physics*, *63*(2-3), 155–169, doi:10.1016/s1364-6826(00)00138-3.

Hocking, W. K. (1999), Temperatures using radar-meteor decay times, *Geophysical Research Letters*, *26*(21), 3297–3300, doi:10.1029/1999GL003618.

Hocking, W. K., T. Thayaparan, and J. Jones (1997), Meteor decay times and their use in determining a diagnostic mesospheric Temperature-pressure parameter: Methodology and one year of data, *Geophysical Research Letters*, *24*(23), 2977–2980, doi:10.1029/97GL03048.

Hocking, W. K., R. E. Silber, J. M. C. Plane, W. Feng, and M. Garbanzo-Salas (2016), Decay times of transitionally dense specularly reflecting meteor trails and potential chemical impact on trail lifetimes, *Ann. Geophys*, *34*, 1119–1144, doi:10.5194/angeo-34-1119-2016.

Holdsworth, D. A., I. M. Reid, and M. A. Cervera (2004), Buckland Park all-sky interferometric meteor radar, *Radio Science*, *39*(5), n/a–n/a, doi:10.1029/2003RS003014.

Jones, J., B. McIntosh, and M. Simek (1990), Ozone and the duration of overdense radio meteors, *Journal of Atmospheric and Terrestrial Physics*, *52*(4), 253–258, doi:10.1016/0021-9169(90)90092-2.

Jones, J., A. R. Webster, and W. K. Hocking (1998), An improved interferometer design for use with meteor radars, *Radio Science*, *33*(1), 55–65, doi:10.1029/97RS03050.

Jones, W. (1995), The decay of radar echoes from meteors with particular reference to their use in the determination of temperature fluctuations near the mesopause, *Annales Geophysicae*, *13*(10), 1104–1106, doi:10.1007/s00585-995-1104-x.

Jones, W., and J. Jones (1990), Ionic diffusion in meteor trains, *Journal of Atmospheric and Terrestrial Physics*, *52*(3), 185–191, doi:10.1016/0021-9169(90)90122-4.

Kaiser, T. (1953), Radio echo studies of meteor ionization, *Advances in Physics*, *2*(8), 495–544, doi:10.1080/00018735300101282.

Kim, J.-H., Y. H. Kim, G. Jee, and C. Lee (2012), Mesospheric temperature estimation from meteor decay times of weak and strong meteor trails, *Journal of Atmospheric and Solar-Terrestrial Physics*, *89*, 18–26, doi:10.1016/J.JASTP.2012.07.003.

Kowalczyk, A. (2017), *Support Vector Machines Succinctly*, 1 ed.

Kozlovsky, A., R. Lukianova, S. Shalimov, and M. Lester (2016), Mesospheric temperature estimation from meteor decay times during Geminids meteor shower, *Journal of Geophysical Research: Space Physics*, *121*(2), 1669–1679, doi:10.1002/2015JA022222.

Mawrey, R. S., and A. D. Broadhurst (1993), Modeling the effect of meteor radiant distributions on the detection rate of meteor signals over forward-scatter links, *Radio Science*, *28*(3), 428–440, doi:10.1029/93RS00319.

McKinley, D. (1961), *Meteor Science and Engineering*, McGraw-Hill.

Mitchell, N. J., D. Pancheva, H. R. Middleton, and M. E. Hagan (2002), Mean winds and tides in the Arctic mesosphere and lower thermosphere, *Journal of Geophysical Research: Space Physics*, *107*(A1), 1004, doi:10.1029/2001JA900127.

Plane, J. M. C. (2003), Atmospheric Chemistry of Meteoric Metals, doi:10.1021/CR0205309.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1992), *Numerical Recipes in C*, 2nd ed., Cambridge University Press, Cambridge.

Strelnikova, I., M. Rapp, S. Raizada, and M. Sulzer (2007), Meteor smoke particle properties derived from Arecibo incoherent scatter radar observations, *Geophysical Research Letters*, *34*(15), doi:10.1029/2007GL030635.

Thomas, R. J., C. A. Barth, and S. Solomon (1984), Seasonal variations of ozone in the upper mesosphere and gravity waves, *Geophysical Research Letters*, *11*(7), 673–676, doi: 10.1029/GL011i007p00673.

Vaudrin, C. V., S. E. Palo, and J. L. Chau (2018), Complex Plane Specular Meteor Radar Interferometry, *Radio Science*, *53*(1), 112–128, doi:10.1002/2017RS006317.

Younger, J. P., I. M. Reid, R. A. Vincent, and D. J. Murphy (2015), A method for estimating the height of a mesospheric density level using meteor radar, *Geophysical Research Letters*, *42*(14), doi:10.1002/2015GL065066.

Zhao, S., J. Urbina, L. Dyrud, and R. Seal (2011), Multilayer detection and classification of specular and nonspecular meteor trails, *Radio Science*, *46*(6), doi:10.1029/2010RS004548.

Appendix A: Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm allows for a model that depends nonlinearly on a set of M unknown parameters $a_k, k = 1, 2, ..., M$ to be fit to a set of observational data. Due to the nonlinear dependencies on the model parameters, the fitting algorithm is applied iteratively, until a best fit is found. The treatment of the Levenberg-Marquardt algorithm in this thesis is summarized from *Press et al.* [1992]. The model to be fitted to a set of data points (x_i, y_i) is given in Equation A.1.

$$y = y(x; \mathbf{a}) \tag{A.1}$$

A merit function χ^2 is defined in Equation A.2 such that the global minimum of the merit function corresponds to the set of model parameters that best fit the observed data. Given a trial set of parameters, the the goal is to iteratively minimize χ^2 and update the trial parameters until χ^2 is sufficiently minimized, at which point the optimized model parameters are known.

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i; \mathbf{a})}{\sigma_i} \right]^2 \tag{A.2}$$

The gradient of χ^2 provides information about which direction each of the trial parameters needs to move in order to further minimize χ^2. The gradient of χ^2 is defined in Equation A.3.

$$\frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i=1}^{N} \frac{[y_i - y(x_i; \mathbf{a})]}{\sigma_i^2} \frac{\partial y(x_i; \mathbf{a})}{\partial a_k} \qquad k = 1, 2, ..., M \tag{A.3}$$

An additional partial derivative of χ^2 is defined in Equation A.4 and provides information about how steeply χ^2 changes with respect to each trial parameter, which is needed to determine an appropriate step size for the minimization algorithm.

$$\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i; \mathbf{a})}{\partial a_k} \frac{\partial y(x_i; \mathbf{a})}{\partial a_l} - [y_i - y(x_i; \mathbf{a})] \frac{\partial^2 y(x_i; \mathbf{a})}{\partial a_l \partial a_k} \right] \tag{A.4}$$

It is common to remove the factors of 2 by defining β_k and α_{kl}, as in Equation A.5. α_{kl} is equal to one-half times the Hessian and usually referred to as the curvature matrix.

$$\beta_k \equiv -\frac{1}{2}\frac{\partial \chi^2}{\partial a_k} \qquad \alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \tag{A.5}$$

β_k and α_{kl} can be rewritten as a set of linear equations, as in Equation A.6, where δa_l are the increments that will be added to the trial parameters for the next iteration of the algorithm.

$$\sum_{i=1}^{M} \alpha_{kl} \delta a_l = \beta_k \tag{A.6}$$

This set of linear equations is solved with a general solution given in Equation A.7.

$$\delta a_l = \text{constant} \times \beta_l \tag{A.7}$$

To allow for variable step sizes in the minimization of χ^2, a compensation factor λ is introduced, shown in Equation A.8. Ideally, the result will be small steps when the curvature of χ^2 is steep and larger steps when the curvature is shallow.

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l \qquad \text{or} \qquad \lambda \alpha_{ll} \delta a_l = \beta_l \tag{A.8}$$

Equations A.8 and A.6 can be combined if a new matrix α' is introduced, as defined in Equations A.9 and A.10.

$$\alpha'_{jj} \equiv a_{jj}(1 + \lambda) \qquad (j = k) \tag{A.9}$$

$$\alpha'_{jk} \equiv a_{jk} \qquad (j \neq k) \tag{A.10}$$

Using the α' matrix, Equations A.8 and A.6 can be replaced by Equation A.10.

$$\sum_{l=1}^{M} \alpha'_{kl} \delta a_l = \beta_k \tag{A.11}$$

If λ is very large, α' becomes diagonally dominant, so that Equation A.10 approaches Equation A.8. Conversely, if λ is very small, Equation A.10 approaches Equation A.6 instead. For a given set of initial trial parameters, the flow of the Levenberg-Marquardt algorithm is as follows:

1. Compute $\chi^2(\mathbf{a})$.
2. Pick a small value for λ, such as 0.001.
3. Solve Equation A.10 for $\delta\mathbf{a}$ and evaluate $\chi^2(\mathbf{a}+\delta\mathbf{a})$.
4. If $\chi^2(\mathbf{a}+\delta\mathbf{a}) \geq \chi^2(\mathbf{a})$, increase λ by a factor of 10 and go back to step 3.
5. If $\chi^2(\mathbf{a}+\delta\mathbf{a}) < \chi^2(\mathbf{a})$, decrease λ by a factor of 10, update the trial solution such that $\mathbf{a} = \mathbf{a}+\delta\mathbf{a}$, and go back to step 3.

This procedure should continue until χ^2 decreases by less than some small amount, usually on the order of 10^{-2} or 10^{-3}. Iterating to complete convergence within the machine roundoff limit is often unnecessary and for most applications will not increase the goodness of the model fit since the model parameters that minimize χ^2 are at best a statistical estimate of the true parameters.

After χ^2 has been minimized, the covariance matrix of the standard errors for the fitted parameters can be found by computing the matrix in Equation A.11.

$$[C] \equiv [\alpha]^{-1} \tag{A.12}$$

Appendix B: Poker Flat Meteor Radar Level 1 Data Product

The standard Level 1 data product of the processing software developed for this thesis includes radar detections for one day. The file is stored with a .pfmr1 file extension, and is an ASCII plain text file. The contents of the file are divided into three sections: underdense meteors, possible overdense meteors, and other detections. There is a file header that contains some information about the configuration of the radar at the time of data collection. These files are generated by analyzing the CEV (confirmed event) files that are generated by the Genesis software for each radar detection. CEV files contain 4 seconds of raw radar data for detections that are classified as underdense meteors by the Skicorr processing. Table B.1 contains information on the data fields included in the PFMR1 files. Data fields that have a statistical uncertainty associated with them will have a '+/-' field containing that uncertainty.

Table B.1: Format of PFMR Level 1 data files.

Data Field	**Description**
DATE	The date of the detection (UTC).
TIME	The time of the detection (UTC).
FILE	The file extension for the CEV file associated with this detection.
RANGE	The range of the detection in kilometers.
HEIGHT1	The estimated height of the detection in kilometers.
HEIGHT2	The estimated height, using nonlinear least squares fitting.
VRAD	The estimated Doppler velocity of the detected object in m/s.
ZEN1	The estimated zenith angle, using Holdsworth interferometry, in degrees.
AZI1	The estimated azimuth angle, using Holdsworth interferometry, in degrees. Zero degrees corresponds to east, ninety degrees corresponds to north, etc.
ZEN2	The estimated zenith angle, using nonlinear least squares fitting.
AZI2	The estimated azimuth angle, using nonlinear least squares fitting.
DUR	The duration of the detection in seconds.
SNR	The signal to noise ratio of the received signal in dB.
DIFF	The estimated ambipolar diffusion coefficient in m^2/s.
CHISQ	Chi-squared goodness of fit metric from model fit. Sum of the residual over the duration of the echo, divided by the duration of the echo.

```
%LEVEL01DATA.m
%create level 1 data product. Run this script and select dates. A level 1
%data product will be produced for each day that data is available.
clearvars;
clc;
warning('off','all'); %turn off warning display in command window
%setup
prompt = {'Start Date (Y,M,D)','End Date (Y,M,D)','Network drive'};
dlgtitle = 'Setup';
dims = [1 20];
definput = {'2018,12,23','2018,12,24','Y'};
x = inputdlg(prompt,dlgtitle,dims,definput);
dayStart = [str2num(x{1}),0,0,0];
dayStop = [str2num(x{2}),0,0,0];
drv = x{3};
D1 = datenum(datestr(dayStart));
D2 = datenum(datestr(dayStop));
numDays = D2 - D1;
if (numDays>=0)
%add mpd files to path
addpath(genpath([drv,':\PokerFlatMeteorRadar\METEORS']));
for iii = 0:numDays
    day = datevec(D1+iii);
    save('day.mat','day');
    %add directories to path for this day
```

```
if (mod(day(3),2)==0)
    if (day(3)<10) && (day(2)>=10)
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),num2str(day(2)),'\METEOR0',num2str(day(3)-1)]));
    elseif (day(3)>=10) && (day(2)<10)
            if (day(3)~=10)
                    addpath(genpath([drv,':\PokerFlatMeteorRadar\
                        ↪ CEVArchive\',num2str(day(1)),'0',num2str(day(2)
                        ↪ ),'\METEOR',num2str(day(3)-1)]));
            else
                    addpath(genpath([drv,':\PokerFlatMeteorRadar\
                        ↪ CEVArchive\',num2str(day(1)),'0',num2str(day(2)
                        ↪ ),'\METEOR09']));
            end
    elseif (day(3)<10) && (day(2)<10)
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),'0',num2str(day(2)),'\METEOR0',num2str(day(3)-1)
            ↪ ]));
    else
            if (day(3)~=10)
                    addpath(genpath([drv,':\PokerFlatMeteorRadar\
                        ↪ CEVArchive\',num2str(day(1)),num2str(day(2)),'\
                        ↪ METEOR',num2str(day(3)-1)]));
            else
                    addpath(genpath([drv,':\PokerFlatMeteorRadar\
                        ↪ CEVArchive\',num2str(day(1)),num2str(day(2)),'\
                        ↪ METEOR09']));
```

```
            end
    end
else
    if (day(3)<10) && (day(2)>=10)
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),num2str(day(2)),'\METEOR0',num2str(day(3))]));
    elseif (day(3)>=10) && (day(2)<10)
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),'0',num2str(day(2)),'\METEOR',num2str(day(3))]));
    elseif (day(3)<10) && (day(2)<10)
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),'0',num2str(day(2)),'\METEOR0',num2str(day(3))]))
            ↪ ;
    else
        addpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str
            ↪ (day(1)),num2str(day(2)),'\METEOR',num2str(day(3))]));
    end
end
%level 1 processing for this day
run('mpdProcessingStable.m');
%clear variables for next run
clearvars -except D1 D2 numDays iii drv day
%remove folders from path
if (mod(day(3),2)==0)
    if (day(3)<10) && (day(2)>=10)
        rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
            ↪ day(1)),num2str(day(2)),'\METEOR0',num2str(day(3)-1)]));
```

```
    elseif (day(3)>=10) && (day(2)<10)
            if (day(3)~=10)
                    rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive
                        ↪ \',num2str(day(1)),'0',num2str(day(2)),'\METEOR
                        ↪ ',num2str(day(3)-1)]));
            else
                    rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive
                        ↪ \',num2str(day(1)),'0',num2str(day(2)),'\
                        ↪ METEOR09']));
            end
    elseif (day(3)<10) && (day(2)<10)
        rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
            ↪ day(1)),'0',num2str(day(2)),'\METEOR0',num2str(day(3)-1)])
            ↪ );
    else
            if (day(3)~=10)
                    rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive
                        ↪ \',num2str(day(1)),num2str(day(2)),'\METEOR',
                        ↪ num2str(day(3)-1)]));
            else
                    rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive
                        ↪ \',num2str(day(1)),num2str(day(2)),'\METEOR09'
                        ↪ ]));
            end
    end
else
    if (day(3)<10) && (day(2)>=10)
```

```
            rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
                ↪ day(1)),num2str(day(2)),'\METEOR0',num2str(day(3))]));
        elseif (day(3)>=10) && (day(2)<10)
            rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
                ↪ day(1)),'0',num2str(day(2)),'\METEOR',num2str(day(3))]));
        elseif (day(3)<10) && (day(2)<10)
            rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
                ↪ day(1)),'0',num2str(day(2)),'\METEOR0',num2str(day(3))]));
        else
            rmpath(genpath([drv,':\PokerFlatMeteorRadar\CEVArchive\',num2str(
                ↪ day(1)),num2str(day(2)),'\METEOR',num2str(day(3))]));
        end
    end
end
%remove MPD files from path
rmpath(genpath([drv,':\PokerFlatMeteorRadar\METEORS']));
else
        disp('ERROR: Start date must be before end date');
end
```

```
%mpdProcessingStable.m
%Use data from MPD file to check invidual CEV files for underdense echoes.
%creates list of underdense echoes along with some parameters from the MPD
%file.
%process detections at all heights and all times for the day
hLim = [-inf inf];
tLim1 = [0 0 0 0 0 0];
tLim2 = [0 0 0 23 59 59];
load('day.mat');
if (day(3)<10)
    str = ['ME',num2str(day(1)),num2str(day(2)),'0',num2str(day(3))];
else
    str = ['ME',num2str(day(1)),num2str(day(2)),num2str(day(3))];
end
if (day(2)<10)
   str = [str(1:6),'0',str(7:end)];
end
%Load MPD data
%#########################################################################
%
fname=['mp',int2str(sum(day(1:3).*[10000 100 1])),'.pokerflat.mpd'];
datdir=['./METEORS/'];
%datdir=['./ME',fname(3:10),'/'];
%read MPD header
fid = fopen(fname);
if (fid==-1) %file does not exist
```

```
    fprintf('No data for %s-%s-%s ...\n',num2str(day(1)),num2str(day(2)),
        ↪ num2str(day(3)));
else
    %parse mpd file
    fprintf('Parsing MPD file for %s-%s-%s ...\n',num2str(day(1)),num2str(
        ↪ day(2)),num2str(day(3)));
    %read file line by line to extract header info
    tline = fgetl(fid);
    while (~strcmp(tline(1:4),'MODE'))
        tline = fgetl(fid);
        if (length(tline)>=3)
            if (strcmp(tline(1:3),'PRF'))
                PRF = str2num(tline(5:end));
            end
        end
        if (length(tline)>=5)
            if (strcmp(tline(1:5),'GATES'))
                GATES = str2num(tline(7:end));
            end
        end
        if (length(tline)>=6)
            if (strcmp(tline(1:6),'RANGE '))
                RANGE = str2num(tline(7:end));
            end
        end
        if (length(tline)>=7)
            if (strcmp(tline(1:7),'RX_GAIN'))
```

```
        RX_GAIN = str2num(tline(9:end));
    end
end
if (length(tline)>=8)
    if (strcmp(tline(1:8),'CHANNELS'))
        CHANNELS = str2num(tline(10:end));
    end
    if (strcmp(tline(1:8),'LOCATION'))
        LOCATION = str2num(tline(10:end));
    end
    if (strcmp(tline(1:8),'SITENAME'))
        SITENAME = tline(10:end);
    end
end
if (length(tline)>=9)
    if (strcmp(tline(1:9),'FREQUENCY'))
        FREQUENCY = str2num(tline(11:end));
    end
    if (strcmp(tline(1:9),'RX_FILTER'))
        RX_FILTER = str2num(tline(11:end));
    end
end
if (length(tline)>=10)
    if (strcmp(tline(1:10),'PULSE_CODE'))
        PULSE_CODE = str2num(tline(12:end));
    end
    if (strcmp(tline(1:10),'RANGE_ZERO'))
```

```
        RANGE_ZERO = str2num(tline(12:end));
    end
    if (strcmp(tline(1:10),'RESOLUTION'))
        RESOLUTION = str2num(tline(12:end));
    end
end
if (length(tline)>=11)
    if (strcmp(tline(1:11),'PULSE_SHAPE'))
        PULSE_SHAPE = tline(13:end);
    end
end
if (length(tline)>=12)
    if (strcmp(tline(1:12),'CABLE_PHASES'))
        CABLE_PHASES = str2num(tline(14:end));
    end
    if (strcmp(tline(1:12),'PULSE_LENGTH'))
        PULSE_LENGTH = str2num(tline(14:end));
    end
    if (strcmp(tline(1:12),'PULSE_NUMBER'))
        PULSE_NUMBER = str2num(tline(14:end));
    end
    if (strcmp(tline(1:12),'TX_PHASEFLIP'))
        TX_PHASEFLIP = str2num(tline(14:end));
    end
end
if (length(tline)>=13)
    if (strcmp(tline(1:13),'PHASE_OFFSETS'))
```

```
        PHASE_OFFSETS = str2num(tline(15:end));
    end
    if (strcmp(tline(1:13),'START_RGE(KM)'))
        START_RGE = str2num(tline(15:end));
    end
end
if (length(tline)>=14)
    if (strcmp(tline(1:14),'ANTENNA_COORDS'))
        ANTENNA_COORDS = str2num(tline(16:end));
    end
    if (strcmp(tline(1:14),'FREQUENCY(MHz)'))
        FREQUENCY = str2num(tline(16:end));
    end
    if (strcmp(tline(1:14),'RESOLUTION(KM)'))
        RESOLUTION = str2num(tline(16:end));
    end
end
if (length(tline)>=15)
    if (strcmp(tline(1:15),'PULSE_AMPLITUDE'))
        PULSE_AMPLITUDE = str2num(tline(17:end));
    end
    if (strcmp(tline(1:15),'PULSE_RANGEZERO'))
        PULSE_RANGEZERO = str2num(tline(17:end));
    end
    if (strcmp(tline(1:15),'RECEIVER_PHASES'))
        RECEIVER_PHASES = str2num(tline(17:end));
    end
```

```
    end
end
fclose(fid);
% load mpd file
fid = fopen(fname, 'rt');
mpdin = fread(fid);
fclose(fid);
LF = find(mpdin==10);
mpd.Header = char(mpdin(1:LF(29)))'; % Maybe this is useful for
    ↪ something.
Data = reshape(char(mpdin(LF(29)+1:end)), 127, [])';
Data(:,127) = []; % Remove LF
mpd.FileID = Data(:,26:30); % Save File ID for future reference
%mpd.FileName = ['ME20171213.', mpd.FileID];
Data(:,26:31) = []; % Remove File ID
% Define a format string for reading the numerical data
sstr = '␣%u/%u/%u␣%u:%u:%f␣%f␣%f␣%f␣%f␣%f␣%f␣%u␣%f␣%u␣%u␣%f␣%f␣%f␣%f';
% Read all data in one go
aux = sscanf(Data', sstr);
% There are 20 values, make 20-column matrix
val = reshape(aux, 20, [])';
% Columns are as follows:
% Date Time Rge Ht Vrad delVr Theta Phi0 Ambig Delphase
% 1-3 4-6 7 8 9 10 11 12 13 14
% antpair IREX amax mpdTau vmet snrdb
% 15 16 17 18 19 20
mpd.t = val(:,1:6); % Time in y, m, d, mpdHeight, m, s
```

```
% day = val(:,1:3);
mpd.tn = datenum(mpd.t); % Time numerically (Matlab date number)
% Make a date string for plots
mpd.dstr = sprintf('%u-%02u-%02u', mpd.t(1,1:3));
% Range
mpd.rge = val(:,7);
% Height
mpd.h = val(:,8);
% Velocity
mpd.vr = val(:, 9);
mpd.dvr = val(:,10);
% Extract zenith angle; compute elevation
mpd.za = val(:,11);
mpd.el = 90-mpd.za;
% Extract azimuth
% Conversion: E=0, N=90, W=180, S=270 to normal: b=mod(90-a,360);
mpd.az = mod(90-val(:,12), 360);
mpd.az0 = val(:,12);
% Ambiguity
mpd.amb = val(:,13);
% Phase error
mpd.dphi = val(:,14);
% Entrance velocities
mpd.ve = val(:,19);
% Decay times
mpd.mpdTau = val(:,18);
% Antenna pair
```

```
mpd.pair = val(:,15);
% SNR in dB
mpd.snr = val(:,20);
% amax
mpd.amax = val(:,17);
% IREX
mpd.irex = val(:,16);
%
    ↪ ##########################################################################
    ↪
%%
mpdLength = length(mpd.za);
tol = 0.03; %tolerance for determination of a close duplicate
ii0 = 1; %index for meteors that are used in analysis
ii1 = 0; %index for meteors rejected due to being exact duplicates
ii2 = 0; %index for meteors rejected for being close duplicates
duplicate = 0; %is an echoe a duplicate of the previous echo?
overwrite = 0; %overwrite previous entry? (for duplicates)
for i = 1:mpdLength
    mpd.FileName(i,:) = [str,'.', mpd.FileID(i,:)];
    if (i~=1)
        %check for exact duplicates
        if (mpd.h(i)==mpd.h(mpdPrev)) && (mpd.za(i)==mpd.za(mpdPrev)) &&
            ↪ (mpd.az0(i)==mpd.az0(mpdPrev))
            if (mpd.amax(i)>mpd.amax(mpdPrev))
                overwrite = 1;
            else
```

```
        overwrite = 0;
    end
    duplicate = 1;
    ii1 = ii1 + 1;
    doubles(ii1,:) = mpd.FileID(i,:); %list of exact duplicates
else
    overwrite = 0;
    duplicate = 0;
end
%check for close duplicates
%look for echoes that are within 10 ms of previous echo and
   ↪ have
%height, zenith, azimuth within 3% of previous echo value
if (duplicate==0)
    hTol = abs(mpd.h(i)-mpd.h(mpdPrev))/mpd.h(i);
    zTol = abs(mpd.za(i)-mpd.za(mpdPrev))/mpd.za(i);
    aTol = abs(mpd.az0(i)-mpd.az0(mpdPrev))/mpd.az0(i);
    tTol = abs(mpd.t(i,6)-mpd.t(mpdPrev,6));
    if (hTol<=tol) && (zTol<=tol) && (aTol<=tol) && (tTol<=0.010)
        if (mpd.amax(i)>mpd.amax(mpdPrev))
            overwrite = 1;
        else
            overwrite = 0;
        end
        duplicate = 1;
        ii2 = ii2 + 1;
```

```
            closeDoubles(ii2,:) = mpd.FileID(i,:); %list of close
                ↪ duplicates
        end
    end
end
if (overwrite==1)
    duplicate = 0; %allow previous entry to be overwritten
end
if (duplicate==0) && (mpd.amb(i)==1)
    if (mpd.h(i)>=hLim(1)) && (mpd.h(i)<=hLim(2)) %check height
        ↪ restriction
        if (timeCheck(tLim1,tLim2,mpd.t(i,:))) %check time
            ↪ restriction
            if (overwrite==1)
                ii0 = ii0-1; %overwrite previous entry, duplicate
                    ↪ echo
            end
            mpdTau(ii0) = mpd.mpdTau(i); %half-amplitude decay time
            mpdHeight(ii0) = mpd.h(i); %height
            mpdTheta(ii0) = mpd.za(i); %zenith angle
            mpdPhi(ii0) = mpd.az0(i); %azimuth angle
            mpdVr(ii0) = mpd.vr(i); %radial velocity
            mpdTime(ii0,:) = mpd.t(i,1:6); %time of detection
            mpdFiles(ii0,:) = mpd.FileName(i,:); %list of files used
                ↪ in temperature calculation
            ii0 = ii0 + 1;
        end
```

```
        end
    end

    mpdPrev = i; %save current line in MPD
end
fprintf('%i␣duplicate␣objects␣removed␣across␣entire␣day\n',(ii1+ii2));
fprintf('%i␣unambiguous␣objects␣found\n',(ii0-1));
if exist('mpdFiles','var') == 1
    if exist('doubles','var') == 1
        if exist('closedoubles','var') == 1
            if (day(3)<10)
                save([num2str(day(1)),num2str(day(2)),'0',num2str(day(3))
                    ↪ ,'MPDPARSE_RESULTS.mat'],'mpdFiles','mpdVr','mpdTau
                    ↪ ','mpdHeight','mpdTheta', ...
                    'mpdPhi','mpdTime','doubles','closeDoubles','tLim1','
                        ↪ tLim2','hLim','day','SITENAME','LOCATION','
                        ↪ FREQUENCY','CHANNELS','RESOLUTION', ...
                    'GATES','START_RGE','PRF','ANTENNA_COORDS','
                        ↪ PHASE_OFFSETS','PULSE_CODE');
            else
                save([num2str(day(1)),num2str(day(2)),num2str(day(3)),'
                    ↪ MPDPARSE_RESULTS.mat'],'mpdFiles','mpdVr','mpdTau',
                    ↪ 'mpdHeight','mpdTheta', ...
                    'mpdPhi','mpdTime','doubles','closeDoubles','tLim1','
                        ↪ tLim2','hLim','day','SITENAME','LOCATION','
                        ↪ FREQUENCY','CHANNELS','RESOLUTION', ...
```

```
                    'GATES','START_RGE','PRF','ANTENNA_COORDS','
                        ↪ PHASE_OFFSETS','PULSE_CODE');
            end
        else
            if (day(3)<10)
                save([num2str(day(1)),num2str(day(2)),'0',num2str(day(3))
                    ↪ ,'MPDPARSE_RESULTS.mat'],'mpdFiles','mpdVr','mpdTau
                    ↪ ','mpdHeight','mpdTheta', ...
                    'mpdPhi','mpdTime','tLim1','tLim2','hLim','day','
                        ↪ SITENAME','LOCATION','FREQUENCY','CHANNELS','
                        ↪ RESOLUTION','GATES','START_RGE', ...
                    'PRF','ANTENNA_COORDS','PHASE_OFFSETS','PULSE_CODE');
            else
                save([num2str(day(1)),num2str(day(2)),num2str(day(3)),'
                    ↪ MPDPARSE_RESULTS.mat'],'mpdFiles','mpdVr','mpdTau',
                    ↪ 'mpdHeight','mpdTheta', ...
                    'mpdPhi','mpdTime','tLim1','tLim2','hLim','day','
                        ↪ SITENAME','LOCATION','FREQUENCY','CHANNELS','
                        ↪ RESOLUTION','GATES','START_RGE', ...
                    'PRF','ANTENNA_COORDS','PHASE_OFFSETS','PULSE_CODE');
            end
        end
    end
end
%run meteor classification
run('classifStable.m');
%create level 1 data file
```

```
    run('PFMR_datafile.m');
    %create meteor plots
    %run('meteorPlotStable.m');
    %cleanup extra files
    if (day(3)<10)
        delete([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'
            ↪ MPDPARSE_RESULTS.mat']);
        delete([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'
            ↪ UNDERDENSE_RESULTS.mat']);
    else
        delete([num2str(day(1)),num2str(day(2)),num2str(day(3)),'
            ↪ MPDPARSE_RESULTS.mat']);
        delete([num2str(day(1)),num2str(day(2)),num2str(day(3)),'
            ↪ UNDERDENSE_RESULTS.mat']);
    end
end
delete('day.mat');
```

```
%classifStable.m
%Classify underdense meteors
fprintf('Looking for underdense echoes...\n')
fprintf('This may take awhile...\n')
load('day.mat');
%load the list of files to process
if (day(3)<10)
    load([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'
        ↪ MPDPARSE_RESULTS.mat']);
else
    load([num2str(day(1)),num2str(day(2)),num2str(day(3)),'MPDPARSE_RESULTS.
        ↪ mat']);
end
echo_filename = mpdFiles;
numFiles = length(echo_filename);
%start parallel pool if one doesn't already exist
p = gcp;
critCount = zeros(1,17);
load('PFMR_constants.mat');
numPair = 6; %number of antenna pairs
otherCount = 0; %track number of rejected echoes
confirmCount = 0; %count underdense echoes
overCount = 0; %count overdense echoes
azimuth = []; %record azimuth angles for confirmed meteors
zenith = []; %zenith angles for confirmed meteors
eTime = []; %1/e decay times for confirmed meteors
```

```
hTime = []; %half-amplitude decay time for confirmed meteors
mHeight = []; %heights for confirmed meteors
%##########################################################################
for ii = 1:numFiles
    data = readCEV(echo_filename(ii,:)); %read data from file
    if (data.ignore==1)
       continue;
    end
    %keep list of rejected and confirmed meteor echoes, as well as some
    %important parameters for those echoes
    param = underdenseParameters(data); %calculate parameters [fd Da R
       ↪ paramF(9) paramF(10) SNR]
    [theta, phi, H] = ThetaPhiH(param(4),param(5),param(3)); %calculate
       ↪ zenith,azimuth,height
    crit = HoldsworthClassification(data,freq,theta,phi,0,0,H,0);
    critCount = critCount + crit;
    if (crit(1)==1) %confirmed underdense echo
        param = underdenseParameters(data); %calculate parameters [fd Da R
           ↪ paramF(9) paramF(10) SNR]
        [theta1, phi1, H1] = ThetaPhiH(param(4),param(5),param(3));
        a = [1, 1, 1, 1, 1, param(1), param(2), -param(3)*K, param(4), param
           ↪ (5)];
        paramA = a;
        %disp(a);
        iter = 500;
        [paramF,R,covar,covar2] = LevMarMeteorFit(data,a,K,d,gamma,iter,0);
        acc_check = 1;
```

```
if (acc_check)
        [theta2, phi2, H2] = ThetaPhiH(paramF(9),paramF(10),-paramF
            ↪ (8)/K); %calculate zenith,azimuth,height
end
if (imag(theta2)~=0)
    %[theta, phi, H] = ThetaPhiH(param(4),param(5),param(3));
    %paramF(9) = param(4);
    %paramF(10) = param(5);
    continue;
end
if (acc_check) %good fit
        rge = -paramF(8)/K;
        %statistical uncertainty for model parameters and calculated
            ↪ parameters
        SD = 2.576.*sqrt(diag(covar))'; %statistical uncertainty to
            ↪ 99% confidence interval for model parameters
        SD2 = 2.576.*sqrt(diag(covar2))'; %uncerainty for azimuth,
            ↪ zenith, and height
        dx = SD(9); dy = SD(10);
        %propagate uncertainty
        dTheta = SD2(2)*180/pi; %sqrt((dthetay*dy)^2+(dthetax*dx)^2);
        dPhi = SD2(1)*180/pi; %sqrt((dphiy*dy)^2+(dphix*dx)^2);
        dH = SD2(3); %sqrt((dhr*SD(8))^2 + (dhtheta*dTheta)^2)/1e3;
        dVr = sqrt((-c*SD(6)/(2*freq))^2);
        underdense(confirmCount+1,:) = echo_filename(ii,:);
        tau_f(confirmCount+1) = mpdTau(ii);
        h_f(confirmCount+1) = mpdHeight(ii);
```

```
        zen_f(confirmCount+1) = mpdTheta(ii);
        azi_f(confirmCount+1) = mpdPhi(ii);
        detTime_f(confirmCount+1,:) = mpdTime(ii,1:6);
        vr_f(confirmCount+1) = mpdVr(ii);
        tau_c(confirmCount+1) = (lam^2*log(2))/(16*pi^2*paramF(7));
        fd_c(confirmCount+1) = paramF(6);
        zen1_c(confirmCount+1) = theta1;
        azi1_c(confirmCount+1) = phi1;
        h1_c(confirmCount+1) = H1;
        zen2_c(confirmCount+1) = theta2;
        azi2_c(confirmCount+1) = phi2;
        h2_c(confirmCount+1) = H2;
        SNR_c(confirmCount+1) = param(6);
        range_c(confirmCount+1) = rge;
        uncert(confirmCount+1,:) = [SD,dTheta,dPhi,dH,dVr];
        vr_c(confirmCount+1) = data.vrad;
        dur_c(confirmCount+1) = data.dur;
        diff_c(confirmCount+1) = paramF(7);
        chisq(confirmCount+1) = sum(R)/(data.dur/data.IPP);
        if (sum(isnan(SD))~=0) || (sum(isnan(SD2))~=0)
                for ijk = 1:length(uncert(confirmCount+1,:))
                        uncert(confirmCount+1,ijk) = -999;
                end
        end
        confirmCount = confirmCount + 1;
else %bad fit
        rge = param(3);
```

```
dx = -999; dy = -999;
dTheta = -999;
dPhi = -999;
dH = -999;
dVr = -999;
underdense(confirmCount+1,:) = echo_filename(ii,:);
tau_f(confirmCount+1) = mpdTau(ii);
h_f(confirmCount+1) = mpdHeight(ii);
zen_f(confirmCount+1) = mpdTheta(ii);
azi_f(confirmCount+1) = mpdPhi(ii);
detTime_f(confirmCount+1,:) = mpdTime(ii,1:6);
vr_f(confirmCount+1) = mpdVr(ii);
tau_c(confirmCount+1) = data.halfTime;
fd_c = param(1);
zen1_c(confirmCount+1) = theta;
azi1_c(confirmCount+1) = phi;
h1_c(confirmCount+1) = H;
zen2_c(confirmCount+1) = -999;
azi2_c(confirmCount+1) = -999;
h2_c(confirmCount+1) = -999;
SNR_c(confirmCount+1) = param(6);
range_c(confirmCount+1) = rge;
uncert(confirmCount+1,:) = [ones(1,10).*-999, dTheta, dPhi,
    ↪ dH, dVr];
vr_c(confirmCount+1) = data.vrad;
dur_c(confirmCount+1) = data.dur;
if (data.halfTime==-999)
```

```
                diff_c(confirmCount+1) = -999;
            else
                diff_c(confirmCount+1) = (lam^2*log(2))/(16*pi^2*data.
                    ↪ halfTime);
            end
            confirmCount = confirmCount + 1;
    end
elseif (crit(17)==1) %possible overdense echo
    param = underdenseParameters(data); %calculate parameters [fd Da R
        ↪ paramF(9) paramF(10) SNR]
    [theta, phi, H] = ThetaPhiH(param(4),param(5),param(3)); %calculate
        ↪ zenith,azimuth,height
    if (imag(theta)~=0)
        %theta = -999; phi = -999; H = -999;
        continue;
    end
    overdense(overCount+1,:) = echo_filename(ii,:);
    tau_ompd(overCount+1) = mpdTau(ii);
    h_ompd(overCount+1) = mpdHeight(ii);
    zen_ompd(overCount+1) = mpdTheta(ii);
    azi_ompd(overCount+1) = mpdPhi(ii);
    detTime_ompd(overCount+1,:) = mpdTime(ii,1:6);
    vr_ompd(overCount+1) = mpdVr(ii);
    fd_ocalc(overCount+1) = param(1);
    vr_ocalc(overCount+1) = -param(1)*c/(2*freq);
    zen_ocalc(overCount+1) = theta;
    azi_ocalc(overCount+1) = phi;
```

```
        h_ocalc(overCount+1) = H;
        SNR_ocalc(overCount+1) = param(6);
        range_ocalc(overCount+1) = param(3);
        dur_ocalc(overCount+1) = data.dur;
        overCount = overCount + 1;
    else %not an underdense echo
        param = underdenseParameters(data); %calculate parameters [fd Da R
            ↪ paramF(9) paramF(10) SNR]
        [theta, phi, H] = ThetaPhiH(param(4),param(5),param(3)); %calculate
            ↪ zenith,azimuth,height
        if (imag(theta)~=0)
            %theta = -999; phi = -999; H = -999;
            continue;
        end
        otherList(otherCount+1,:) = echo_filename(ii,:);
        detTime_other(otherCount+1,:) = mpdTime(ii,1:6);
        SNR_other(otherCount+1) = param(6);
        zen_other(otherCount+1) = theta;
        azi_other(otherCount+1) = phi;
        h_other(otherCount+1) = H;
        range_other(otherCount+1) = param(3);
        vr_other(otherCount+1) = data.vrad; %-param(1)*c/(2*freq);
        dur_other(otherCount+1) = data.dur;
        otherCount = otherCount + 1;
    end

end
```

```
clear mpdTau mpdHeight mpdTheta mpdPhi mpdTime mpdVr
mpdTau = tau_f; mpdHeight = h_f; mpdTheta = zen_f; mpdPhi = azi_f; mpdTime =
    ↪  detTime_f; mpdVr = vr_f;
fprintf('%i underdense echoes identified\n',length(underdense))
if exist('underdense','var') == 1
    if (day(3)<10)
    save([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'
        ↪ UNDERDENSE_RESULTS.mat'],'underdense','mpdVr','mpdTau','mpdHeight
        ↪ ','mpdTheta', ...
        'mpdPhi','mpdTime','tLim1','tLim2','hLim','day','tau_c','fd_c','
            ↪ zen1_c','azi1_c','h1_c','zen2_c','azi2_c','h2_c','overdense','
            ↪ tau_ompd','h_ompd','zen_ompd', ...
        'azi_ompd','detTime_ompd','vr_ompd','fd_ocalc','zen_ocalc','
            ↪ azi_ocalc','h_ocalc','otherList','detTime_other','SNR_c','
            ↪ SNR_ocalc', ...
        'range_c','range_ocalc','uncert','dur_ocalc','vr_c','dur_c','diff_c
            ↪ ','SNR_other','zen_other','azi_other','h_other','range_other',
            ↪  ...
        'vr_other','dur_other','vr_ocalc','chisq');
    else
    save([num2str(day(1)),num2str(day(2)),num2str(day(3)),'
        ↪ UNDERDENSE_RESULTS.mat'],'underdense','mpdVr','mpdTau','mpdHeight
        ↪ ','mpdTheta', ...
        'mpdPhi','mpdTime','tLim1','tLim2','hLim','day','tau_c','fd_c','
            ↪ zen1_c','azi1_c','h1_c','zen2_c','azi2_c','h2_c','overdense','
            ↪ tau_ompd','h_ompd','zen_ompd', ...
```

```
        'azi_ompd','detTime_ompd','vr_ompd','fd_ocalc','zen_ocalc','
            ↪ azi_ocalc','h_ocalc','otherList','detTime_other','SNR_c','
            ↪ SNR_ocalc', ...
        'range_c','range_ocalc','uncert','dur_ocalc','vr_c','dur_c','diff_c
            ↪ ','SNR_other','zen_other','azi_other','h_other','range_other',
            ↪  ...
        'vr_other','dur_other','vr_ocalc','chisq');
    end
end
```

```
%PFMR_datafile.m

%create level 1 data file

load('day.mat');

if (day(3)<10)

    load([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'

        ↪ MPDPARSE_RESULTS.mat']);

    load([num2str(day(1)),num2str(day(2)),'0',num2str(day(3)),'

        ↪ UNDERDENSE_RESULTS.mat']);

else

    load([num2str(day(1)),num2str(day(2)),num2str(day(3)),'MPDPARSE_RESULTS

        ↪ .mat']);

    load([num2str(day(1)),num2str(day(2)),num2str(day(3)),'

        ↪ UNDERDENSE_RESULTS.mat']);

end

DATE = [num2str(day(1)),'-',num2str(day(2)),'-',num2str(day(3))];

if (day(3)<10) && (day(2)>=10)

    fid = fopen([drv,':\PokerFlatMeteorRadar\DataLevel1\',num2str(day(1)),

        ↪ num2str(day(2)),'0',num2str(day(3)),'.pfmr1'],'w');

elseif (day(3)>=10) && (day(2)<10)

    fid = fopen([drv,':\PokerFlatMeteorRadar\DataLevel1\',num2str(day(1)),'0

        ↪ ',num2str(day(2)),num2str(day(3)),'.pfmr1'],'w');

elseif (day(3)<10) && (day(2)<10)

    fid = fopen([drv,':\PokerFlatMeteorRadar\DataLevel1\',num2str(day(1)),'0

        ↪ ',num2str(day(2)),'0',num2str(day(3)),'.pfmr1'],'w');

else
```

```
    fid = fopen([drv,':\PokerFlatMeteorRadar\DataLevel1\',num2str(day(1)),
        ↪ num2str(day(2)),num2str(day(3)),'.pfmr1'],'w');
end
fprintf(fid,'#PFMR Level 1 Data \r\n');
fprintf(fid,'SOFTWARE VERSION 1.1\r\n');
fprintf(fid,'SITENAME %15s \r\n',SITENAME);
fprintf(fid,'LOCATION %12s \r\n',num2str(LOCATION));
fprintf(fid,'FREQUENCY %f \r\n',FREQUENCY);
fprintf(fid,'CHANNELS %f \r\n',CHANNELS);
fprintf(fid,'RESOLUTION %f \r\n',RESOLUTION);
fprintf(fid,'GATES %f \r\n',GATES);
fprintf(fid,'START_RANGE %f \r\n',START_RGE);
fprintf(fid,'PRF %f \r\n',PRF);
fprintf(fid,'ANTENNA_COORDS %50s \r\n',num2str(ANTENNA_COORDS));
fprintf(fid,'PHASE_OFFSETS %50s \r\n',num2str(PHASE_OFFSETS));
fprintf(fid,'PULSE_CODE %d \r\n',PULSE_CODE);
fprintf(fid,'UNDERDENSE \r\n');
fprintf(fid,'%6s %12s %10s %7s %7s %7s %7s %7s %7s %7s %7s %7s %7s %7s
    ↪ %7s %7s %7s %7s %7s \r\n','DATE','TIME','FILE','RANGE','+/-','HT1','
    ↪ HT2','+/-','VRAD','ZEN1','AZI1','ZEN2','+/-','AZI2','+/-','DUR','SNR'
    ↪ ,'DIFF','+/-','CHISQ');
for i = 1:length(underdense)
    if (mpdTime(i,4)<10)
        HOUR = ['0',num2str(mpdTime(i,4))];
    else
        HOUR = num2str(mpdTime(i,4));
    end
```

```
if (mpdTime(i,5)<10)
    MIN = ['0',num2str(mpdTime(i,5))];
else
    MIN = num2str(mpdTime(i,5));
end
if (mpdTime(i,6)<10)
    SEC = ['0',num2str(mpdTime(i,6))];
else
    SEC = num2str(mpdTime(i,6));
end
if ((mpdTime(i,6)-floor(mpdTime(i,6)))==0)
   SEC = [SEC,'.000'];
end
TIME = [HOUR,':',MIN,':',SEC];
while (length(TIME)<12)
   TIME = [TIME,'0'];
end
FILE = underdense(i,12:end);
fprintf(fid,'%10s␣%11s␣%6s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣
    ↪ %7s␣%7s␣%7s␣%7s␣%7s␣%7s␣\r\n' ...
    ,DATE,TIME,FILE,num2str(round(range_c(i),1)),num2str(round(uncert(i
       ↪ ,8),1)),num2str(round(h1_c(i),1)),num2str(round(h2_c(i),1)),
       ↪ num2str(round(uncert(i,13),1)), ...
    num2str(round(vr_c(i),1)),num2str(round(zen1_c(i),1)),num2str(round(
       ↪ azi1_c(i),1)),num2str(round(zen2_c(i),1)),num2str(round(uncert
       ↪ (i,11),1)),num2str(round(azi2_c(i),1)), ...
```

```
        num2str(round(uncert(i,12),1)),num2str(round(dur_c(i),1)),num2str(
            ↪ round(SNR_c(i),1)),num2str(round(diff_c(i),1)),num2str(round(
            ↪ uncert(i,7),1)),num2str(round(chisq(i),3)));
end
fprintf(fid,'OVERDENSE \r\n');
fprintf(fid,'%6s %12s %10s %7s %7s %7s %7s %7s %7s \r\n','DATE','TIME',
    ↪ 'FILE','RANGE','HEIGHT','VRAD','ZEN','AZI','DUR','SNR');
%fprintf(fid,'\r\n');
for i = 1:length(overdense)
    if (detTime_ompd(i,4)<10)
        HOUR = ['0',num2str(detTime_ompd(i,4))];
    else
        HOUR = num2str(detTime_ompd(i,4));
    end
    if (detTime_ompd(i,5)<10)
        MIN = ['0',num2str(detTime_ompd(i,5))];
    else
        MIN = num2str(detTime_ompd(i,5));
    end
    if (detTime_ompd(i,6)<10)
        SEC = ['0',num2str(detTime_ompd(i,6))];
    else
        SEC = num2str(detTime_ompd(i,6));
    end
    if ((detTime_ompd(i,6)-floor(detTime_ompd(i,6)))==0)
       SEC = [SEC,'.000'];
    end
```

```
    TIME = [HOUR,':',MIN,':',SEC];
    while (length(TIME)<12)
       TIME = [TIME,'0'];
    end
    FILE = overdense(i,12:end);
    fprintf(fid,'%10s␣%11s␣%6s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣\r\n',DATE,TIME,
       ↪ FILE,num2str(round(range_ocalc(i),1)),num2str(round(h_ocalc(i),1)
       ↪ ), ...
        num2str(round(vr_ocalc(i),1)),num2str(round(zen_ocalc(i),1)),num2str
           ↪ (round(azi_ocalc(i),1)) ...
        ,num2str(round(dur_ocalc(i),1)),num2str(round(SNR_ocalc(i),1)));
end
fprintf(fid,'OTHER␣\r\n');
fprintf(fid,'%6s␣%12s␣%10s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣\r\n','DATE','TIME',
   ↪ 'FILE','RANGE','HEIGHT','VRAD','ZEN','AZI','DUR','SNR');
for i = 1:length(otherList)
    if (detTime_other(i,4)<10)
        HOUR = ['0',num2str(detTime_other(i,4))];
    else
        HOUR = num2str(detTime_other(i,4));
    end
    if (detTime_other(i,5)<10)
        MIN = ['0',num2str(detTime_other(i,5))];
    else
        MIN = num2str(detTime_other(i,5));
    end
    if (detTime_other(i,6)<10)
```

```
        SEC = ['0',num2str(detTime_other(i,6))];
    else
        SEC = num2str(detTime_other(i,6));
    end
    if ((detTime_other(i,6)-floor(detTime_other(i,6)))==0)
       SEC = [SEC,'.000'];
    end
    TIME = [HOUR,':',MIN,':',SEC];
    while (length(TIME)<12)
       TIME = [TIME,'0'];
    end
    FILE = otherList(i,12:end);
    fprintf(fid,'%10s␣%11s␣%6s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣%7s␣\r\n',DATE,TIME,
        ↪ FILE,num2str(round(range_other(i),1)),num2str(round(h_other(i),1)
        ↪ ), ...
        num2str(round(vr_other(i),1)),num2str(round(zen_other(i),1)),num2str
            ↪ (round(azi_other(i),1)) ...
        ,num2str(round(dur_other(i),1)),num2str(round(SNR_other(i),1)));
end
fprintf(fid,'END␣\r\n');
fclose(fid);
```

```
%CEVread.m
%readCEV files
function echo = readCEV(filename)
data = GetCEV_0(filename);
N = data.RECL_PTS; %number of data points
echo.rgeGate = data.RGE; %range gate of detection
cpar = data.cpar;
echo.IPP = cpar(1);
lam = 9.2102;
c = physconst('LightSpeed');
offset = [.00 -12.60 -15.50 -13.80 -12.80] .* pi ./ 180; %antenna phase
    ↪ offsets
echo.offset = offset;
echo.ignore = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo.ampr = zeros(5,N);
echo.powr = zeros(5,N);
%echo.powrdb = zeros(5,N);
echo.inph = zeros(5,N);
echo.quadph = zeros(5,N);
echo.s = zeros(5,N);
echo.phase = zeros(5,N);
echo.phDiff = zeros(1,6);
for k=1:5 %read data for all 5 receivers
    eval(['b=data.r',int2str(k),';']) %pull out data for one receiver
    %b(1,:) = I -- b(2,:) = Q
```

```
    echo.inph(k,:) = b(1,:);
    echo.quadph(k,:) = b(2,:);
    echo.s(k,:) = complex(echo.inph(k,:),echo.quadph(k,:)); %complex time
        ↪ series
    echo.ampr(k,:)=sqrt(echo.inph(k,:).^2+echo.quadph(k,:).^2); %total
        ↪ magnitude from I and Q components
    echo.powr(k,:)=echo.inph(k,:).^2+echo.quadph(k,:).^2; %total power from
        ↪  I and Q components
    %echo.powrdb(k,:) = radardb(echo.powr(k,:));
    echo.phase(k,:) = atan(echo.quadph(k,:)./echo.inph(k,:)); %phase from I
        ↪  and Q components
end
%Phase differences
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pair1 = [5 1 1 5 4 3]; %used for interferometry
pair2 = [2 5 2 3 5 4];
for k = 1:length(pair1) %phase difference between each pair
    [pairCCF, pairLags] = xcorr(echo.s(pair2(k),:),echo.s(pair1(k),:));
    center = find(pairLags==0);
    magCCF = abs(pairCCF);
    magCCF2 = [magCCF(center-2), magCCF(center-1), magCCF(center+1), magCCF(
        ↪ center+2)];
    L = [-2, -1, 1, 2];
    magFit = polyfit(L,magCCF2,1);
    phCCF = unwrap(angle(pairCCF));
    phCCF2 = [phCCF(center-2), phCCF(center-1), phCCF(center+1), phCCF(
        ↪ center+2)];
```

```
    phFit = polyfit(L,phCCF2,1);
    echo.phDiff(k) = phFit(2);
    echo.phDiff(k) = echo.phDiff(k) + offset(pair2(k)) - offset(pair1(k));
end
%coherently combine receiver signals%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
timeDelay = round((echo.phDiff .* lam ./ (2*pi) ./ c));
maxDelay = max(abs([max(timeDelay),min(timeDelay)]));
echo.amprSeries = zeros(1,N-maxDelay);
echo.powrSeries = zeros(1,N-maxDelay);
echo.powrdbSeries = zeros(1,N-maxDelay);
echo.complexSeries = zeros(1,N-maxDelay);
for k = (1+maxDelay):(N-maxDelay) %add signals from all receivers, in phase
    echo.amprSeries(k) = echo.ampr(5,k) + echo.ampr(1,k-timeDelay(1)) + echo
        ↪ .ampr(2,k+timeDelay(2)) + echo.ampr(3,k-timeDelay(4)) + echo.ampr
        ↪ (4,k+timeDelay(5));
    echo.powrSeries(k) = echo.powr(5,k) + echo.powr(1,k-timeDelay(1)) + echo
        ↪ .powr(2,k+timeDelay(2)) + echo.powr(3,k-timeDelay(4)) + echo.powr
        ↪ (4,k+timeDelay(5));
    echo.complexSeries(k) = echo.s(5,k) + echo.s(1,k-timeDelay(1)) + echo.s
        ↪ (2,k+timeDelay(2)) + echo.s(3,k-timeDelay(4)) + echo.s(4,k+
        ↪ timeDelay(5));
    %echo.powrSeries(k) = echo.powr(5,k) + echo.powr(1,k) + echo.powr(2,k)
        ↪ + echo.powr(3,k) + echo.powr(4,k);
end
echo.amprSeries = echo.amprSeries(find(echo.amprSeries,1,'first'):find(echo
    ↪ .amprSeries,1,'last'));
```

```
echo.powrSeries = echo.powrSeries(find(echo.powrSeries,1,'first'):find(echo
    ↪ .powrSeries,1,'last'));
echo.complexSeries = echo.complexSeries(find(echo.complexSeries,1,'first'):
    ↪ find(echo.complexSeries,1,'last'));
echo.powrdbSeries = radardb(echo.powrSeries);
echo.N = N;
echo.Nc = length(echo.amprSeries);
echo.Ndb = length(echo.powrdbSeries);
%isolate echo
pwrdbSmooth = smooth(echo.powrdbSeries,5)';
maxPwr = max(pwrdbSmooth); %max power in db
maxPwrI = find(pwrdbSmooth==maxPwr); %index of max power
minPwr = min(pwrdbSmooth); %min power in db
minPwrI = find(pwrdbSmooth==minPwr); %index of min power
thresh = mean(pwrdbSmooth);
%find start and end of echo, find where signal is below threshold for 10ms
buff = floor(50e-3/echo.IPP);
echoEnd = maxPwrI + min((find(pwrdbSmooth(maxPwrI:end)<thresh))) - 1;
echoBegin = max(find(pwrdbSmooth(1:maxPwrI)<thresh));
if (isempty(echoEnd))
    echoEnd = echo.Ndb;
end
if (isempty(echoBegin))
    echoBegin = 1;
end
%check to make sure signal is below threshold for 10 ms on both ends
if (echoEnd~=echo.Ndb) && (echoBegin~=1)
```

```
check = 1;
checkEnd = 0; checkBegin = 0;
addEnd = 1; addBegin = 1;
if (echoBegin<=buff)
   echoBegin = 1;
   checkBegin = 1;
end
if (echoEnd>=echo.Ndb-buff)
   echoEnd = echo.Ndb;
   checkEnd = 1;
end
while (check)
    if (~checkEnd)
        if (all(pwrdbSmooth(echoEnd:echoEnd+buff)<thresh))
            checkEnd = 1;
        else
            echoEnd = maxPwrI + min((find(pwrdbSmooth(maxPwrI:end)<thresh
                ↪ ))) + addEnd;
            addEnd = addEnd + 1;
            if (echoEnd>=echo.Ndb-buff)
                echoEnd = echo.Ndb;
                checkEnd = 1;
            end
        end
    end
    if(~checkBegin)
        %disp(echoBegin-buff);
```

```
            if (all(pwrdbSmooth(echoBegin-buff:echoBegin)<thresh))
                checkBegin = 1;
            else
                echoBegin = max(find(pwrdbSmooth(1:maxPwrI)<thresh)) -
                    ↪ addBegin;
                addBegin = addBegin + 1;
                if (echoBegin<=buff)
                    echoBegin = 1;
                    checkBegin = 1;
                end
            end
        end
        if (checkEnd) && (checkBegin)
           break
        end
    end
end
dur = (echoEnd-echoBegin) * echo.IPP; %echo duration in seconds
echo.begin = echoBegin;
echo.end = echoEnd;
echo.peak = maxPwrI;
echo.dur = dur;
B = echoBegin;%maxPwrI;%echoBegin;
E = echoEnd;
if (B==E)
   B = 1;
   E = 5;
```

```
   echo.ignore = 1;
end
if (abs(B-E)<6)
   B = ceil(length(pwrdbSmooth)/2);
   E = B + 10;
   echo.ignore = 1;
end
if (isempty(B))
   B = 1;
   E = 5;
   echo.ignore = 1;
end
if (isnan(B))
   B = 1;
   E = 5;
   echo.ignore = 1;
end
if (isempty(E))
   B = 1;
   E = 5;
   echo.ignore = 1;
end
if (isnan(E))
   B = 1;
   E = 5;
   echo.ignore = 1;
end
```

```
echo.begin = B;
echo.end = E;
echo.dur = (E - B) * echo.IPP;
%Phase differences (using echo start and end times)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CCFtotal = zeros(1,4);
for k = 1:length(pair1) %phase difference between each pair
    [pairCCF, pairLags] = xcorr(echo.s(pair2(k),B:E),echo.s(pair1(k),B:E));
    center = find(pairLags==0);
    magCCF = abs(pairCCF);
    magCCF2 = [magCCF(center-2), magCCF(center-1), magCCF(center+1), magCCF(
        ↪ center+2)];
    L = [-2, -1, 1, 2];
    magFit = polyfit(L,magCCF2,1);
    CCFpeak = magFit(2);
    phCCF = unwrap(angle(pairCCF));
    phCCF2 = [phCCF(center-2), phCCF(center-1), phCCF(center+1), phCCF(
        ↪ center+2)];
    phFit = polyfit(L,phCCF2,1);
    echo.phDiff(k) = phFit(2);
    echo.phDiff(k) = echo.phDiff(k) + offset(pair2(k)) - offset(pair1(k));
    CCFtotal = CCFtotal + [pairCCF(center-2), pairCCF(center-1), pairCCF(
        ↪ center+1), pairCCF(center+2)];
    if (k==3)
            halfSample = find(abs(pairCCF(center:length(pairCCF)))<=(CCFpeak
                ↪ /2));
            if (isempty(halfSample))
```

```
                echo.halfTime = -999;
            else
                echo.halfTime = (halfSample(1)+1) * echo.IPP;
            end
    end
end
phCCFtotal = unwrap(angle(CCFtotal));
phFit = polyfit(L,phCCFtotal,1);
echo.vrad = - lam * (1/phFit(1)) / (2*pi); %radial velocity
end
```

```
%HoldsworthClassification.m
%Use classification for underdense meteors from [Holdsworth 2004].
%Takes data struct from readCEV.m along with:
%frequency, zenith, azimuth, height, and ambiguousness parameters
%
%Rejection criteria: (contents of variable 'crit')
%1 - no error, analysis ok
%2 - SNR too low
%3 - Ambiguous AOA
%4 - AOA not feasible
%5 - Large difference in AOA from different antenna baselines
%6 - Echo at start or end of time series
%7 - Echo less than 5 samples long
%8 - Echo rise exceeds 0.3 seconds
%9 - Echo decay time less than 2x rise time
%10 - Large power level before echo
%11 - Large power level after echo
%12 - Poor fit to amplitude for decay time estimation
%13 - Poor fit to CCF phase variation
%14 - Height unresolvable
%15 - Ambiguous height
%16 - Radial drift velocity greater than 200 m/s
%17 - Oscillatory echo, could be overdense meteor
function crit = HoldsworthClassification(data,f0,theta,phi,err21,err43,H,
    ↪ ambigH)
c = physconst('LightSpeed'); %speed of light (m/data.s)
```

```
crit = ones(1,17);
crit(1) = 0;
data.powrdbSeries = smooth(data.powrdbSeries,5)';
data.amprSeries = smooth(data.amprSeries,5)';
data.powrSeries = smooth(data.powrSeries,5)';
R = smooth(real(data.complexSeries),5)';
I = smooth(imag(data.complexSeries),5)';
maxPwr = max(data.powrdbSeries); %max power in db
maxPwrI = find(data.powrdbSeries==maxPwr); %index of max power
noise = mean(data.powrdbSeries)-3; %noise level for CEV file
%thresh = noise + 3; %3db above noise floor
minPwr = min(data.powrdbSeries); %min power in db
minPwrI = find(data.powrdbSeries==minPwr);
thresh = mean(data.powrdbSeries);
%find start and end of echo
if (data.end>=length(data.complexSeries))
        data.end = length(data.complexSeries);
end
echoEnd = data.end; %maxPwrI + min((find(data.powrdbSeries(maxPwrI:end)<
    ↪ thresh))) - 1;
echoBegin = data.begin; %max(find(data.powrdbSeries(1:maxPwrI)<thresh));
if (isempty(echoEnd))
    echoEnd = length(data.powrSeries)-10;
end
if (isempty(echoBegin))
    echoBegin = 1;
end
```

```
dur = (echoEnd-echoBegin) * data.IPP; %echo duration in seconds
%SNR
maxSNR = 10*log10(max(data.powrSeries)/mean(data.powrSeries));
echoInd = round(linspace(echoBegin,echoEnd,10));
echoPoints = data.powrSeries(echoInd) ./ max(data.powrSeries);
sb = floor(10e-3/data.IPP);
%average of 8 normalized power data before echo
if (echoBegin>sb)
    before = mean(data.powrSeries(echoBegin-sb:echoBegin)) ./ max(data.
        ↪ powrSeries);
else
    before = mean(data.powrSeries(1:7)) ./ max(data.powrSeries);
end
%average of 8 normalized power data after echo
if (echoEnd<data.Nc-sb)
    after = mean(data.powrSeries(echoEnd:echoEnd+sb)) ./ max(data.powrSeries
        ↪ );
else
    after = mean(data.powrSeries(data.Nc-sb:end)) ./ max(data.powrSeries);
end
%doppler frequency from fft
Fs = 1/data.IPP;
L = length(maxPwrI:echoEnd);
Y = fft(data.complexSeries(maxPwrI:echoEnd));
P2 = abs(Y/L);
f2 = Fs*(0:L-1)/L;
fd = f2(find(P2==max(P2)));
```

```
vd = -fd*c/(2*f0);
powrnorm = data.powrSeries./max(data.powrSeries);
%check from peak to end of signal
ind01 = min(find(powrnorm(maxPwrI:echoEnd)<0.3)) + maxPwrI - 1;
ind02 = min(find(powrnorm(ind01:echoEnd)>0.7)) + ind01 - 1;
%check from peak to beginning of signal
ind03 = max(find(powrnorm(echoBegin:maxPwrI)<0.3)) + echoBegin - 1;
ind04 = max(find(powrnorm(echoBegin:ind03)>0.7)) + echoBegin - 1;
if (isempty(ind02)) && (isempty(ind04))
       crit(17) = 0;
else
       crit(17) = 1;
end
%check criteria for underdense meteor
if (maxSNR>=4)
    crit(2) = 0;
end
if (imag(theta)==0)
   crit(3) = 0;
end
if (theta<85)
   crit(4) = 0;
end
crit(5) = 0; %ignore
if (echoBegin>5) && (echoEnd<data.Ndb-5)
    crit(6) = 0;
end
```

```
if (dur>=(data.IPP*5))
    crit(7) = 0;
end
echoRise = (maxPwrI - echoBegin) * data.IPP;
    crit(8) = 0;
echoDecay = (echoEnd - maxPwrI) * data.IPP;
if (1)%(echoDecay>echoRise)
    crit(9) = 0;
end
if (before<0.5)
    crit(10) = 0;
end
if (after<0.5)
    crit(11) = 0;
end
crit(12) = 0; %ignore
crit(13) = 0;
crit(14) = 0;
crit(15) = 0;
    crit(16) = 0;
if (sum(crit)==0)
    crit(1) = 1; %underdense meteor
end
end
```

```
%LevMarMeteorFit.m
%Uses built in MATLAB optimimization tools
%Fit parameters to meteor backscatter using Levenberg-Marquardt non-linear
%least squares method. Takes data struct from readCEV.m
%initial guesses for parameters
% 1 2 3 4 5 6 7 8 9 10
%parameter vector [A1 A2 A3 A4 A5 f_d D_a phi5 theta_x theta_y]
%opt==0 == Levenberg Marquardt algorithm
%opt==1 == Trust Region Reflective algorithm
function [param,residual,covar,covar2] = LevMarMeteorFit(data,a,K,d,gamma,
    ↪ iter,opt)
B = find(real(data.s(1,:))==max(real(data.s(1,:))));%data.peak;%data.begin;
E = data.Ndb;%data.end;
Y = [smooth(real(data.s(1,B:E)),5)'; smooth(imag(data.s(1,B:E)),5)';
    smooth(real(data.s(2,B:E)),5)'; smooth(imag(data.s(2,B:E)),5)';
    smooth(real(data.s(3,B:E)),5)'; smooth(imag(data.s(3,B:E)),5)';
    smooth(real(data.s(4,B:E)),5)'; smooth(imag(data.s(4,B:E)),5)';
    smooth(real(data.s(5,B:E)),5)'; smooth(imag(data.s(5,B:E)),5)']'; %
        ↪ measured signal
t = linspace(0,length(data.s(1,B:E))*data.IPP,length(data.s(1,B:E)));
N = length(t);
sigma_m = diag(ones(1,10));
Amax = max(max(abs(Y)));
for i = 1:10
   Y(:,i) = Y(:,i) ./ Amax;
end
```

```
save('data_temp.mat','Y','t','K','d','gamma','sigma_m');
%options = optimoptions('lsqnonlin','Display','off','Algorithm','levenberg
    ↪ -marquardt','StepTolerance',1e-6,'MaxIterations',iter,'
    ↪ MaxFunctionEvaluations',1e4,'UseParallel',true);
if (opt==0) %levenberg marquardt algorithm
        options = optimoptions('lsqnonlin','Display','off','Algorithm','
            ↪ levenberg-marquardt','ScaleProblem','jacobian','StepTolerance'
            ↪ ,eps,'FunctionTolerance',0.01,'MaxIterations',iter,'
            ↪ MaxFunctionEvaluations',1e4,'UseParallel',true,'
            ↪ SpecifyObjectiveGradient',true);
        [param,resnorm,residual,exitflag,output,L,J] = lsqnonlin(
            ↪ @NLSObjective,a,[],[],options);
elseif (opt==1) %trust region reflective algorithm
        ub = [1, 1, 1, 1, 1, inf, inf, inf, 1, 1];
        lb = [1, 1, 1, 1, 1, -inf, -inf, -inf, -1, -1];
        options = optimoptions('lsqnonlin','Display','iter-detailed','
            ↪ Algorithm','trust-region-reflective','StepTolerance',1e-9,'
            ↪ FunctionTolerance',0.01,'MaxIterations',iter,'
            ↪ MaxFunctionEvaluations',1e4,'UseParallel',true,'
            ↪ SpecifyObjectiveGradient',true);
        [param,resnorm,residual,exitflag,output,L,J] = lsqnonlin(
            ↪ @NLSObjective,a,lb,ub,options);
end
A = length(J);
B = length(param);
covar = sum(residual) .* inv(J'*J) ./ (A-B); %covariance matrix
%propagate uncertainty to height, zenith, azimuth
```

```
%use method in Vaudrin 2018
dx = param(9);
dy = param(10);
R = param(8)/-K;
var1 = diag(covar);
sigma = [var1(9), sqrt(var1(9)*var1(10)), sqrt(var1(9)*var1(8))
        sqrt(var1(10)*var1(9)), var1(10), sqrt(var1(10)*var1(8))
        sqrt(var1(8)*var1(9)), sqrt(var1(8)*var1(10)), var1(8)];
df1x = -dy/(dx^2+dy^2);
df1y = dx/(dx^2+dy^2);
df1r = 0;
df2x = -(sqrt((dy^2/dx^2)+1)-(dy^2/(dx^2*sqrt((dy^2/dx^2)+1))))/(sqrt(1-dx
   ↪ ^2*((dy^2/dx^2)+1)));
df2y = -dy/(dx*sqrt((dy^2/dx^2)+1)*sqrt(1-dx^2*((dy^2/dx^2)+1)));
df2r = 0;
df3x = -(R*dx)/(sqrt(1-dx^2-dy^2));
df3y = -(R*dy)/(sqrt(1-dx^2-dy^2));
df3r = sqrt(1-dx^2-dy^2);
Js = [df1x, df1y, df1r
     df2x, df2y, df2r
     df3x, df3y, df3r];
 covar2 = Js*sigma*Js';
delete data_temp.mat
end
```

```
%meteorModelJacobian.m
%determine Jacobian matix for complex meteor radar interferometry.
%takes model parameters, a, and measured data, Y, time vector, t,
%wavenumber, K = 2*pi/lambda, antenna distance vector, d,
%antenna distance vector, d, antenna angle vector, gamma

function J = meteorModelJacobian(a,Y,t,K,d,gamma)
    N = length(t);
    %Antenna 1 in-phase channel (equation 1)
    %#########################################################################
    k = 1;
    CH = 1; %channel number
    psi = a(8) + 2.*pi.*a(6).*t - K.*d(k).*(a(9).*cos(gamma(k))+a(10).*sin(
        ↪ gamma(k)));
    %first derivatives
    df1(:,1) = a(k).*exp(-2.*a(7).*t).*cos(2.*psi)+a(k).*exp(-2.*a(7).*t)
        ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*cos(psi);
    df1(:,2) = zeros(N,1);
    df1(:,3) = zeros(N,1);
    df1(:,4) = zeros(N,1);
    df1(:,5) = zeros(N,1);
    df1(:,6) = -2.*a(k).^2.*pi.*t.*exp(-2.*a(7).*t).*sin(2.*psi)+4.*pi.*t.*Y
        ↪ (:,CH)'.*a(k).*exp(-a(7).*t).*sin(psi);
    df1(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t).*cos(2.*psi)-a(k).^2.*t.*exp
        ↪ (-2.*a(7).*t)+2.*t.*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos(psi);
```

```
df1(:,8) = -a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)+2.*Y(:,CH)'.*a(k).*
    ↪ exp(-a(7).*t).*sin(psi);
df1(:,9) = a(k).^2.*K.*d(k).*exp(-2.*a(7).*t).*cos(gamma(k)).*sin(2.*psi
    ↪ )-2.*Y(:,CH)'.*a(k).*K.*d(k).*cos(gamma(k)).*exp(-a(7).*t).*sin(
    ↪ psi);
df1(:,10) = a(k).^2.*K.*d(k).*sin(gamma(k)).*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)-2.*Y(:,CH)'.*a(k).*K.*d(k).*sin(gamma(k)).*exp(-a(7).*t).*sin
    ↪ (psi);

%Antenna 1 quadrature channel (equation 2)
%########################################################################
%first derivatives
CH = 2;
df2(:,1) = a(k).*exp(-2.*a(7).*t)-a(k).*exp(-2.*a(7).*t).*cos(2.*psi)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*sin(psi);
df2(:,2) = zeros(N,1);
df2(:,3) = zeros(N,1);
df2(:,4) = zeros(N,1);
df2(:,5) = zeros(N,1);
df2(:,6) = 2.*pi.*t.*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-4.*pi.*Y(:,
    ↪ CH)'.*a(k).*t.*exp(-a(7).*t).*cos(psi);
df2(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t)+a(k).^2.*t.*exp(-2.*a(7).*t).*
    ↪ cos(2.*psi)+2.*Y(:,CH)'.*a(k).*t.*exp(-a(7).*t).*sin(psi);
df2(:,8) = a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-2.*Y(:,CH)'.*a(k).*exp
    ↪ (-a(7).*t).*cos(psi);
```

```
df2(:,9) = -K.*d(k).*cos(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*cos(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);
df2(:,10) = -K.*d(k).*sin(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*sin(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);

%Antenna 2 in-phase channel (equation 3)
%########################################################################
k = 2;
CH = 3;
psi = a(8) + 2.*pi.*a(6).*t - K.*d(k).*(a(9).*cos(gamma(k))+a(10).*sin(
    ↪ gamma(k)));
%first derivatives
df3(:,1) = zeros(N,1);
df3(:,2) = a(k).*exp(-2.*a(7).*t).*cos(2.*psi)+a(k).*exp(-2.*a(7).*t)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*cos(psi);
df3(:,3) = zeros(N,1);
df3(:,4) = zeros(N,1);
df3(:,5) = zeros(N,1);
df3(:,6) = -2.*a(k).^2.*pi.*t.*exp(-2.*a(7).*t).*sin(2.*psi)+4.*pi.*t.*Y
    ↪ (:,CH)'.*a(k).*exp(-a(7).*t).*sin(psi);
df3(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t).*cos(2.*psi)-a(k).^2.*t.*exp
    ↪ (-2.*a(7).*t)+2.*t.*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos(psi);
df3(:,8) = -a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)+2.*Y(:,CH)'.*a(k).*
    ↪ exp(-a(7).*t).*sin(psi);
```

```
df3(:,9) = a(k).^2.*K.*d(k).*exp(-2.*a(7).*t).*cos(gamma(k)).*sin(2.*psi
    ↪ )-2.*Y(:,CH)'.*a(k).*K.*d(k).*cos(gamma(k)).*exp(-a(7).*t).*sin(
    ↪ psi);
df3(:,10) = a(k).^2.*K.*d(k).*sin(gamma(k)).*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)-2.*Y(:,CH)'.*a(k).*K.*d(k).*sin(gamma(k)).*exp(-a(7).*t).*sin
    ↪ (psi);


%Antenna 2 quadrature channel (equation 4)
%########################################################################
CH = 4;
%first derivatives
df4(:,1) = zeros(N,1);
df4(:,2) = a(k).*exp(-2.*a(7).*t)-a(k).*exp(-2.*a(7).*t).*cos(2.*psi)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*sin(psi);
df4(:,3) = zeros(N,1);
df4(:,4) = zeros(N,1);
df4(:,5) = zeros(N,1);
df4(:,6) = 2.*pi.*t.*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-4.*pi.*Y(:,
    ↪ CH)'.*a(k).*t.*exp(-a(7).*t).*cos(psi);
df4(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t)+a(k).^2.*t.*exp(-2.*a(7).*t).*
    ↪ cos(2.*psi)+2.*Y(:,CH)'.*a(k).*t.*exp(-a(7).*t).*sin(psi);
df4(:,8) = a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-2.*Y(:,CH)'.*a(k).*exp
    ↪ (-a(7).*t).*cos(psi);
df4(:,9) = -K.*d(k).*cos(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*cos(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);
```

```
df4(:,10) = -K.*d(k).*sin(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*sin(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);


%Antenna 3 in-phase channel (equation 5)
%##########################################################################
k = 3;
CH = 5;
psi = a(8) + 2.*pi.*a(6).*t - K.*d(k).*(a(9).*cos(gamma(k))+a(10).*sin(
    ↪ gamma(k)));
%first derivatives
df5(:,1) = zeros(N,1);
df5(:,2) = zeros(N,1);
df5(:,3) = a(k).*exp(-2.*a(7).*t).*cos(2.*psi)+a(k).*exp(-2.*a(7).*t)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*cos(psi);
df5(:,4) = zeros(N,1);
df5(:,5) = zeros(N,1);
df5(:,6) = -2.*a(k).^2.*pi.*t.*exp(-2.*a(7).*t).*sin(2.*psi)+4.*pi.*t.*Y
    ↪ (:,CH)'.*a(k).*exp(-a(7).*t).*sin(psi);
df5(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t).*cos(2.*psi)-a(k).^2.*t.*exp
    ↪ (-2.*a(7).*t)+2.*t.*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos(psi);
df5(:,8) = -a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)+2.*Y(:,CH)'.*a(k).*
    ↪ exp(-a(7).*t).*sin(psi);
df5(:,9) = a(k).^2.*K.*d(k).*exp(-2.*a(7).*t).*cos(gamma(k)).*sin(2.*psi
    ↪ )-2.*Y(:,CH)'.*a(k).*K.*d(k).*cos(gamma(k)).*exp(-a(7).*t).*sin(
    ↪ psi);
```

```
df5(:,10) = a(k).^2.*K.*d(k).*sin(gamma(k)).*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)-2.*Y(:,CH)'.*a(k).*K.*d(k).*sin(gamma(k)).*exp(-a(7).*t).*sin
    ↪ (psi);


%Antenna 3 quadrature channel (equation 6)
%########################################################################
CH = 6;
%first derivatives
df6(:,1) = zeros(N,1);
df6(:,2) = zeros(N,1);
df6(:,3) = a(k).*exp(-2.*a(7).*t)-a(k).*exp(-2.*a(7).*t).*cos(2.*psi)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*sin(psi);
df6(:,4) = zeros(N,1);
df6(:,5) = zeros(N,1);
df6(:,6) = 2.*pi.*t.*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-4.*pi.*Y(:,
    ↪ CH)'.*a(k).*t.*exp(-a(7).*t).*cos(psi);
df6(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t)+a(k).^2.*t.*exp(-2.*a(7).*t).*
    ↪ cos(2.*psi)+2.*Y(:,CH)'.*a(k).*t.*exp(-a(7).*t).*sin(psi);
df6(:,8) = a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-2.*Y(:,CH)'.*a(k).*exp
    ↪ (-a(7).*t).*cos(psi);
df6(:,9) = -K.*d(k).*cos(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*cos(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);
df6(:,10) = -K.*d(k).*sin(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)+2.*K.*d(k).*sin(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
    ↪ (psi);
```

```
%Antenna 4 in-phase channel (equation 7)
%########################################################################
k = 4;
CH = 7;
psi = a(8) + 2.*pi.*a(6).*t - K.*d(k).*(a(9).*cos(gamma(k))+a(10).*sin(
    ↪ gamma(k)));
%first derivatives
df7(:,1) = zeros(N,1);
df7(:,2) = zeros(N,1);
df7(:,3) = zeros(N,1);
df7(:,4) = a(k).*exp(-2.*a(7).*t).*cos(2.*psi)+a(k).*exp(-2.*a(7).*t)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*cos(psi);
df7(:,5) = zeros(N,1);
df7(:,6) = -2.*a(k).^2.*pi.*t.*exp(-2.*a(7).*t).*sin(2.*psi)+4.*pi.*t.*Y
    ↪ (:,CH)'.*a(k).*exp(-a(7).*t).*sin(psi);
df7(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t).*cos(2.*psi)-a(k).^2.*t.*exp
    ↪ (-2.*a(7).*t)+2.*t.*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos(psi);
df7(:,8) = -a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)+2.*Y(:,CH)'.*a(k).*
    ↪ exp(-a(7).*t).*sin(psi);
df7(:,9) = a(k).^2.*K.*d(k).*exp(-2.*a(7).*t).*cos(gamma(k)).*sin(2.*psi
    ↪ )-2.*Y(:,CH)'.*a(k).*K.*d(k).*cos(gamma(k)).*exp(-a(7).*t).*sin(
    ↪ psi);
df7(:,10) = a(k).^2.*K.*d(k).*sin(gamma(k)).*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)-2.*Y(:,CH)'.*a(k).*K.*d(k).*sin(gamma(k)).*exp(-a(7).*t).*sin
    ↪ (psi);

%Antenna 4 quadrature channel (equation 8)
```

```
%#########################################################################
CH = 8;
%first derivatives
df8(:,1) = zeros(N,1);
df8(:,2) = zeros(N,1);
df8(:,3) = zeros(N,1);
df8(:,4) = a(k).*exp(-2.*a(7).*t)-a(k).*exp(-2.*a(7).*t).*cos(2.*psi)
   ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*sin(psi);
df8(:,5) = zeros(N,1);
df8(:,6) = 2.*pi.*t.*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-4.*pi.*Y(:,
   ↪ CH)'.*a(k).*t.*exp(-a(7).*t).*cos(psi);
df8(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t)+a(k).^2.*t.*exp(-2.*a(7).*t).*
   ↪ cos(2.*psi)+2.*Y(:,CH)'.*a(k).*t.*exp(-a(7).*t).*sin(psi);
df8(:,8) = a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-2.*Y(:,CH)'.*a(k).*exp
   ↪ (-a(7).*t).*cos(psi);
df8(:,9) = -K.*d(k).*cos(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
   ↪ psi)+2.*K.*d(k).*cos(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
   ↪ (psi);
df8(:,10) = -K.*d(k).*sin(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
   ↪ psi)+2.*K.*d(k).*sin(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
   ↪ (psi);

%Antenna 5 in-phase channel (equation 9)
%#########################################################################
k = 5;
CH = 9;
```

```
psi = a(8) + 2.*pi.*a(6).*t - K.*d(k).*(a(9).*cos(gamma(k))+a(10).*sin(
    ↪ gamma(k)));
%first derivatives
df9(:,1) = zeros(N,1);
df9(:,2) = zeros(N,1);
df9(:,3) = zeros(N,1);
df9(:,4) = zeros(N,1);
df9(:,5) = a(k).*exp(-2.*a(7).*t).*cos(2.*psi)+a(k).*exp(-2.*a(7).*t)
    ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*cos(psi);
df9(:,6) = -2.*a(k).^2.*pi.*t.*exp(-2.*a(7).*t).*sin(2.*psi)+4.*pi.*t.*Y
    ↪ (:,CH)'.*a(k).*exp(-a(7).*t).*sin(psi);
df9(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t).*cos(2.*psi)-a(k).^2.*t.*exp
    ↪ (-2.*a(7).*t)+2.*t.*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos(psi);
df9(:,8) = -a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)+2.*Y(:,CH)'.*a(k).*
    ↪ exp(-a(7).*t).*sin(psi);
df9(:,9) = a(k).^2.*K.*d(k).*exp(-2.*a(7).*t).*cos(gamma(k)).*sin(2.*psi
    ↪ )-2.*Y(:,CH)'.*a(k).*K.*d(k).*cos(gamma(k)).*exp(-a(7).*t).*sin(
    ↪ psi);
df9(:,10) = a(k).^2.*K.*d(k).*sin(gamma(k)).*exp(-2.*a(7).*t).*sin(2.*
    ↪ psi)-2.*Y(:,CH)'.*a(k).*K.*d(k).*sin(gamma(k)).*exp(-a(7).*t).*sin
    ↪ (psi);


%Antenna 5 quadrature channel (equation 10)
%#########################################################################
CH = 10;
%first derivatives
df10(:,1) = zeros(N,1);
```

```
    df10(:,2) = zeros(N,1);
    df10(:,3) = zeros(N,1);
    df10(:,4) = zeros(N,1);
    df10(:,5) = a(k).*exp(-2.*a(7).*t)-a(k).*exp(-2.*a(7).*t).*cos(2.*psi)
        ↪ -2.*Y(:,CH)'.*exp(-a(7).*t).*sin(psi);
    df10(:,6) = 2.*pi.*t.*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-4.*pi.*Y(:,
        ↪ CH)'.*a(k).*t.*exp(-a(7).*t).*cos(psi);
    df10(:,7) = -a(k).^2.*t.*exp(-2.*a(7).*t)+a(k).^2.*t.*exp(-2.*a(7).*t).*
        ↪ cos(2.*psi)+2.*Y(:,CH)'.*a(k).*t.*exp(-a(7).*t).*sin(psi);
    df10(:,8) = a(k).^2.*exp(-2.*a(7).*t).*sin(2.*psi)-2.*Y(:,CH)'.*a(k).*
        ↪ exp(-a(7).*t).*cos(psi);
    df10(:,9) = -K.*d(k).*cos(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
        ↪ psi)+2.*K.*d(k).*cos(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
        ↪ (psi);
    df10(:,10) = -K.*d(k).*sin(gamma(k)).*a(k).^2.*exp(-2.*a(7).*t).*sin(2.*
        ↪ psi)+2.*K.*d(k).*sin(gamma(k)).*Y(:,CH)'.*a(k).*exp(-a(7).*t).*cos
        ↪ (psi);



    %J = [df1; df2; df3; df4; df5; df6; df7; df8; df9; df10];
    J = df1 + df2 + df3 + df4 + df5 + df6 + df7 + df8 + df9 + df10;
end
```

```
%radardb.m
%converts power data from PFMR to DB based on max possible RX power
function DB = radardb(P)
maxPwr = 2*5*(32676^2);
P(find(P==0)) = [];
DB = 10.*log10(P./maxPwr);
end
```

```
%ThetaPhiH.m
%calculate zenith angle, azimuth angle, and height
function [theta, phi, H] = ThetaPhiH(theta_x,theta_y,R)
load('PFMR_constants.mat'); %load constants
vCos = sqrt(1-theta_y^2-theta_x^2); %sqrt(abs(1-dCos43^2-dCos21^2)); %
    ↪ direction cosine in vertical direction
theta = acos(vCos) * 180/pi;
dCos21 = theta_x; dCos43 = theta_y;
if (dCos21>0 && dCos43>0)
    phi = mod(atan2(dCos43,dCos21)*180/pi,360);
elseif (dCos21<0 && dCos43>0)
    phi = mod(atan2(dCos43,dCos21)*180/pi,360);
elseif (dCos21>0 && dCos43<0)
    phi = mod(atan2(dCos43,dCos21)*180/pi+360,360);
elseif (dCos21<0 && dCos43<0)
    phi = mod(atan2(dCos43,dCos21)*180/pi+360,360);
else
    if (dCos21==0 && dCos43>0)
        phi = 90;
    end
    if (dCos21==0 && dCos43<0)
        phi = 270;
    end
    if (dCos43==0 && dCos21>0)
        phi = 0;
    end
```

```
    if (dCos43==0 && dCos21<0)
        phi = 180;
    end
end
phi = phi + 90; %baselines swiched, rotate 90 degrees
phi = mod(phi,360);
%height
H = sqrt(Re^2+R+2*R*Re*cosd(theta))-Re;
end
```

```
%underdenseParameters.m
%estimate parameters for underdense meteors: doppler frequency, diffusion
%coefficient, range, direction cosines. Takes data from readCEV.m
function param = underdenseParameters(data)
load('PFMR_constants.mat'); %load constants
%smooth data
data.powrdbSeries = smooth(data.powrdbSeries,5)';
data.amprSeries = smooth(data.amprSeries,5)';
data.powrSeries = smooth(data.powrSeries,5)';
R = smooth(real(data.complexSeries),5)';
I = smooth(imag(data.complexSeries),5)';
maxPwr = max(data.powrdbSeries); %max power in db
maxPwrI = find(data.powrdbSeries==maxPwr); %index of max power
noise = mean(data.powrdbSeries)-3; %noise level for CEV file
echoEnd = data.end;
echoBegin = data.begin;
dur = data.dur;
maxSNR = 10*log10(max(data.powrSeries)/mean(data.powrSeries));
%find 1/e decay time
amprSeries = data.amprSeries - min(data.amprSeries);
maxAmpr = max(amprSeries);
amprSeries = amprSeries./maxAmpr;
maxAmpr = max(amprSeries);
maxAmprI = find(amprSeries==maxAmpr);
%1/2 decay time found by cross-correlation
pair1 = [5 1 1 5 4 3]; %used for interferometry
```

```
pair2 = [2 5 2 3 5 4];
CCFtotal = zeros(1,4);
for k = 3:3:6%1:length(pair1) %phase difference between each pair
        [pairCCF, pairLags] = xcorr(data.s(pair2(k),:),data.s(pair1(k),:));
        center = find(pairLags==0);
        magCCF = abs(pairCCF);
        magCCF2 = [magCCF(center-2), magCCF(center-1), magCCF(center+1),
            ↪ magCCF(center+2)];
        L = [-2, -1, 1, 2];
        magFit = polyfit(L,magCCF2,1);
        CCFpeak = magFit(2);
        phCCF = unwrap(angle(pairCCF));
        phCCF2 = [phCCF(center-2), phCCF(center-1), phCCF(center+1), phCCF(
            ↪ center+2)];
        phFit = polyfit(L,phCCF2,1);
        phDiff(k) = phFit(2); %angle(pairCCF(center));%phFit(2);
        phDiff(k) = phDiff(k) + offset(pair2(k)) - offset(pair1(k));
        CCFtotal = CCFtotal + [pairCCF(center-2), pairCCF(center-1), pairCCF
            ↪ (center+1), pairCCF(center+2)];
    if (k==3)
        halfSample = find(abs(pairCCF(center:length(pairCCF)))<=(CCFpeak/2))
            ↪ ;%(max(abs(pairCCF(center:length(pairCCF))))/2));
        if (isempty(halfSample))
            halfTime1 = -9.99; %bad value
        else
            halfTime1 = (halfSample(1)+1) * data.IPP;
        end
```

```
    end
    if (k==6)
        halfSample = find(abs(pairCCF(center:length(pairCCF)))<=(CCFpeak/2))
            ↪ ;%(max(abs(pairCCF(center:length(pairCCF))))/2));
        if (isempty(halfSample))
            halfTime2 = -9.99; %bad value
        else
            halfTime2 = (halfSample(1)+1) * data.IPP;
        end
    end
end
if (halfTime1==-9.99)
    tau_e = halfTime2;
    Da = (lam^2*log(2))/(16*pi^2*tau_e);
elseif (halfTime2==-9.99)
    tau_e = halfTime1;
    Da = (lam^2*log(2))/(16*pi^2*tau_e);
elseif (halfTime1==-9.99) && (halfTime2==-9.99)
    Da = -9.99; %bad value
else
    tau_e = (halfTime1+halfTime2)/2;
    Da = (lam^2*log(2))/(16*pi^2*tau_e);
end
%doppler frequency from fft
Fs = 1/data.IPP;
L = length(maxPwrI:length(data.complexSeries));
%Y = fft(data.complexSeries(maxPwrI:echoEnd));
```

```
if (maxPwrI<length(data.complexSeries))
       Y = fft(real(data.complexSeries(maxPwrI:end)));
else
       Y = fft(real(data.complexSeries));
       L = length(data.complexSeries);
end
P2 = abs(Y/L);
f2 = Fs*(0:L-1)/L;
fd = f2(find(P2==max(P2)));
if (length(fd>1))
   fd = fd(1);
end
%find direction cosine estimates
%AOA estimation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Baseline 4-3
halfLam = pi2pi(data.phDiff(4) - data.phDiff(5)); %half lambda phase
    ↪ difference
fullLam = pi2pi(-data.phDiff(6)); %pi2pi(phDiff(5) + phDiff(4)); %4.5
    ↪ lambda phase difference
mult = -4:4;
for mm = 1:9
    angArray43(mm) = fullLam/(9*pi) + mult(mm)*(2/9); %2pi multiples of 4.5
       ↪ lambda estimation
end
[base_error43, AOA_Ind] = min(abs(angArray43 - (halfLam/pi)));
ang43 = angArray43(AOA_Ind); %direction cosine for this baseline
```

```
%Baseline 2-1
halfLam = pi2pi(data.phDiff(2) - data.phDiff(1)); %half lambda phase
    ↪ difference
fullLam = pi2pi(data.phDiff(3)); %pi2pi(phDiff(2) + phDiff(1)); %4.5 lambda
    ↪  phase difference
for mm = 1:9
    angArray21(mm) = fullLam/(9*pi) + mult(mm)*(2/9); %2pi multiples of 4.5
        ↪ lambda estimation
end
[base_error21, AOA_Ind] = min(abs(angArray21 - (halfLam/pi)));
ang21 = angArray21(AOA_Ind); %direction cosine for this baseline
theta_x = ang21; theta_y = ang43;
vCos = sqrt(1-theta_y^2-theta_x^2); %sqrt(abs(1-dCos43^2-dCos21^2)); %
    ↪ direction cosine in vertical direction
theta = acos(vCos) * 180/pi;
%theta = 59.2;
%find range
R = 14 + 1.5*data.rgeGate;
PRF = 1/data.IPP;
Ramb = c/(2*PRF*1e3);
Rmax = 520; %max range based on 110 km height and zenith angle of 80
    ↪ degrees
Nh = round(Rmax/Ramb); %number of ranges in ensemble
for nhi = 1:Nh
    Ri(nhi) = R + (nhi-1)*Ramb;
end
Hi = sqrt(Re.^2+Ri.^2+2.*Re.*Ri.*cosd(theta)) - Re;
```

```
H2 = Hi(find(Hi<=120));
R2 = Ri(find(Hi<=120));
R_final = R2(find(H2>=70));
if (isempty(R_final))
    R_final = Ri(1);
end
%return parameter vector
if (isempty(fd))
   fd = 0;
end
if (isempty(Da))
   Da = 0;
end
if (isempty(R_final))
   R_final = 0;
end
if (isempty(ang21))
   ang21 = 0;
end
if (isempty(ang43))
   ang43 = 0;
end
if (isempty(maxSNR))
   maxSNR = 1;
end
param = [fd, Da, R_final, ang21, ang43, maxSNR];
end
```

```
%readPFMR1.m
%read level 1 data products (.pfmr1).
%returns a struct with all data from the text file.
%make sure directory with data files is included in path.
function data = readPFMR1(filename)
fid = fopen(filename);
data.FILENAME = filename;
tline = fgetl(fid); %throw away first line
%read file header
while (~strcmp(tline(1:10),'UNDERDENSE'))
    tline = fgetl(fid);
    if (length(tline)>=3)
        if (strcmp(tline(1:3),'PRF'))
            data.PRF = str2num(tline(5:end));
        end
    end
    if (length(tline)>=5)
        if (strcmp(tline(1:5),'GATES'))
            data.GATES = str2num(tline(7:end));
        end
    end
    if (length(tline)>=8)
        if (strcmp(tline(1:8),'SITENAME'))
            data.SITENAME = tline(10:end);
        end
        if (strcmp(tline(1:8),'LOCATION'))
```

```
        data.LOCATION = str2num(tline(10:end));
    end
    if (strcmp(tline(1:8),'CHANNELS'))
        data.CHANNELS = str2num(tline(10:end));
    end
end
if (length(tline)>=9)
    if (strcmp(tline(1:9),'FREQUENCY'))
        data.FREQUENCY = str2num(tline(11:end));
    end
end
if (length(tline)>=10)
    if (strcmp(tline(1:10),'RESOLUTION'))
        data.RESOLUTION = str2num(tline(12:end));
    end
    if (strcmp(tline(1:10),'PULSE_CODE'))
        data.PULSE_CODE = str2num(tline(12:end));
    end
end
if (length(tline)>=11)
    if (strcmp(tline(1:11),'START_RANGE'))
        data.START_RANGE = str2num(tline(13:end));
    end
end
if (length(tline)>=13)
    if (strcmp(tline(1:13),'PHASE_OFFSETS'))
        data.PHASE_OFFSETS = str2num(tline(15:end));
```

```
        end
    end
    if (length(tline)>=14)
        if (strcmp(tline(1:14),'ANTENNA_COORDS'))
            data.ANTENNA_COORDS = str2num(tline(16:end));
        end
    end
end
%read underdense data
tline = fgetl(fid);
formatSpec = '%f-%f-%f␣%f:%f:%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f
    ↪ ␣%f␣%f␣%f';
i = 1;
while (~strcmp(tline(1:9),'OVERDENSE'))
    tline = fgetl(fid);
    if (~strcmp(tline(1:9),'OVERDENSE'))
        data.und_fileID(i,:) = tline(26:30); %CEV file ID
        tline(26:30) = []; %remove file ID information from scanned line
        vec = sscanf(tline,formatSpec);
        if (length(vec)==23)
        data.und_time(i) = vec(4) + vec(5)/60 + vec(6)/3600; %detection time
        data.und_range(i) = vec(7); %range
        data.und_rangeE(i) = vec(8); %range uncertainty
        data.und_height1(i) = vec(9); %height 1
        data.und_height2(i) = vec(10); %height 2
        data.und_heightE(i) = vec(11); %height2 uncertainty
        data.und_vrad(i) = vec(12); %radial velocity
```

```
        %data.und_vradE(i) = vec(12); %radial velocity uncertainty
        data.und_zenith1(i) = vec(13); %zenith 1
        data.und_azimuth1(i) = vec(14); %azimuth 1
        data.und_zenith2(i) = vec(15); %zenith 2
        data.und_zenithE(i) = vec(16); %zenith2 uncertainty
        data.und_azimuth2(i) = vec(17); %azimuth
        data.und_azimuthE(i) = vec(18); %azimuth2 uncertainty
        data.und_dur(i) = vec(19); %echo duration
        data.und_snr(i) = vec(20); %SNR
        data.und_diff(i) = vec(21); %diffusion coefficient
        data.und_diffE(i) = vec(22); %diffusion coefficient uncertainty
        data.chisq(i) = vec(23); %model fit chi-squared
            i = i + 1;
        end
    end
end
%read overdense data
tline = fgetl(fid); %column headers
formatSpec = '%f-%f-%f␣%f:%f:%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f';
i = 1;
while (~strcmp(tline(1:5),'OTHER'))
    tline = fgetl(fid);
    if (~strcmp(tline(1:5),'OTHER'))
        data.ovr_fileID(i,:) = tline(26:30); %CEV file ID
        tline(26:30) = []; %remove file ID from scanned line
        vec = sscanf(tline,formatSpec);
        data.ovr_time(i) = vec(4) + vec(5)/60 + vec(6)/3600; %detection time
```

```
        data.ovr_range(i) = vec(7); %range
        data.ovr_height(i) = vec(8); %height
        data.ovr_vrad(i) = vec(9); %radial velocity
        data.ovr_zenith(i) = vec(10); %zenith angle
        data.ovr_azimuth(i) = vec(11); %azimuth angle
        data.ovr_dur(i) = vec(12); %echo duration
        data.ovr_snr(i) = vec(13); %SNR
        i = i + 1;
    end
end
%read other data
tline = fgetl(fid); %column headers
i = 1;
while (~strcmp(tline(1:3),'END'))
   tline = fgetl(fid);
   if (~strcmp(tline(1:3),'END'))
      data.oth_fileID(i,:) = tline(26:30); %CEV file ID
      tline(26:30) = []; %remove file ID from scanned line
      vec = sscanf(tline,formatSpec);
      data.oth_time(i) = vec(4) + vec(5)/60 + vec(6)/3600; %detection time
      data.oth_range(i) = vec(7); %range
      data.oth_height(i) = vec(8); %height
      data.oth_vrad(i) = vec(9); %radial velocity
      data.oth_zenith(i) = vec(10); %zenith angle
      data.oth_azimuth(i) = vec(11); %azimuth angle
      data.oth_dur(i) = vec(12); %echo duration
      data.oth_snr(i) = vec(13); %SNR
```

```
      i = i + 1;
   end
end
fclose(fid);
end
```